Swift: Ready for Primetime?

Chris Bailey(@Chris__Bailey)
Swift @ IBM Engineering Team

January 21st, 2017

Swift Programming

Safe | Expressive | Fast

Hello World

```
print("Hello, world!")

let event = ”Swift Meetup"
print("Hello, \(event)")

event = "Somewhere else" // ERROR

var variableEvent = ”Swift Meetup"
```

Control Flow

```swift
let expenseCosts = [34.4, 30.99, 250.0]

var sum: Double = 0


for expense in expenseCosts {
    sum += expense
}

print("total cost is \(sum)")
```

Switch

```swift
let flavour = "Vanilla"

switch flavour {
    case "Chocolate":
        print("Quite nice")
    case "Strawberry", "Rum'n'raisin":
        print("Very nice")
    case let str where str.hasPrefix("Mint"):
        print("UGH!!!, I hate \(str)")
    default:
        print("No opinion about this")
}
```

Optionals

```swift
var name: String? = "Joe Bloggs"
if let validName = name {
    print("Hello, \(validName)")
} else {
    print("Anonymous, eh...")
}

let str = "42"
let num = Int(str)
if num != nil {
    print("Conversion successful – num is \(num!)")
}

if let num = Int(str) {
    print("Conversion successful – num is \(num)")
} else {
    print("Conversion failed")
}
```

Optional Chaining

```
if variable.myOptional != nil {
    if variable.myOptional!.anotherOptional != nil {
        print(variable.myOptional!.anotherOptional!.item)
    }
}


if let result = variable.myOptional?.anotherOptional?.item {
    print(result)
}
```

Functions

```
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)

func addInts(_ a: Int, _ b: Int) -> Int {
    return a + b
}
addInts(1, 3)


func move(from start: Point, to end: Point) -> Bool { /* code */ }

move(from: a, to: b)
```

Varargs

```swift
func max(numbers: Int...) -> Int {
    var max = numbers[0]
    for number in numbers {
        if number > max {
            max = number
        }
    }
    return max
}

max(1, 2, 3, 4, 5  // 5
max() // ERROR
```

Tuples

```swift
func minAndMax(numbers: Int...) -> (min: Int, max: Int) {
    var min = numbers[0]
    var max = numbers[0]
    for number in numbers {
        if number > max {
            max = number
        } else if number < min {
            min = number
        }
    }
    return (min, max)
}

let result = minAndMax(1, 2, 3, 4, 5)
print(result.min)
print(result.max)
```

Closures

```swift
let numbers = [1, 2, 3]

numbers.map({
    (number: Int) -> Int in
    return number * 5
})
// [5, 10, 15]

numbers.map({
    number in number * 5
})

numbers.map { $0 * 5 }
```

Structs

```
struct Point {
    var x: Int
    var y: Int
  func description() -> String {
        return "x=\(x), y=\(y)"
    }
}

var coord = Point(x: 2, y: 4)

var newCoord = coord
coord.x = 4

print coord.description()    // x=4, y=4
print newCoord.description() // x=2, y=4
```

Enums

```
enum ApprovalStatus {
    case PendingOn(String)
    case Denied
    case Approved(String)
}


var status = ApprovalStatus.PendingOn("Joe Bloggs")

status = .Approved("13213-4341321-2")

switch status {
case .PendingOn(let approver):
    print("Request pending on approval from \(approver)")
case .Denied:
    print("Request DENIED")
case .Approved(let code):
    print("Request approved - auth code \(code)")
}
```

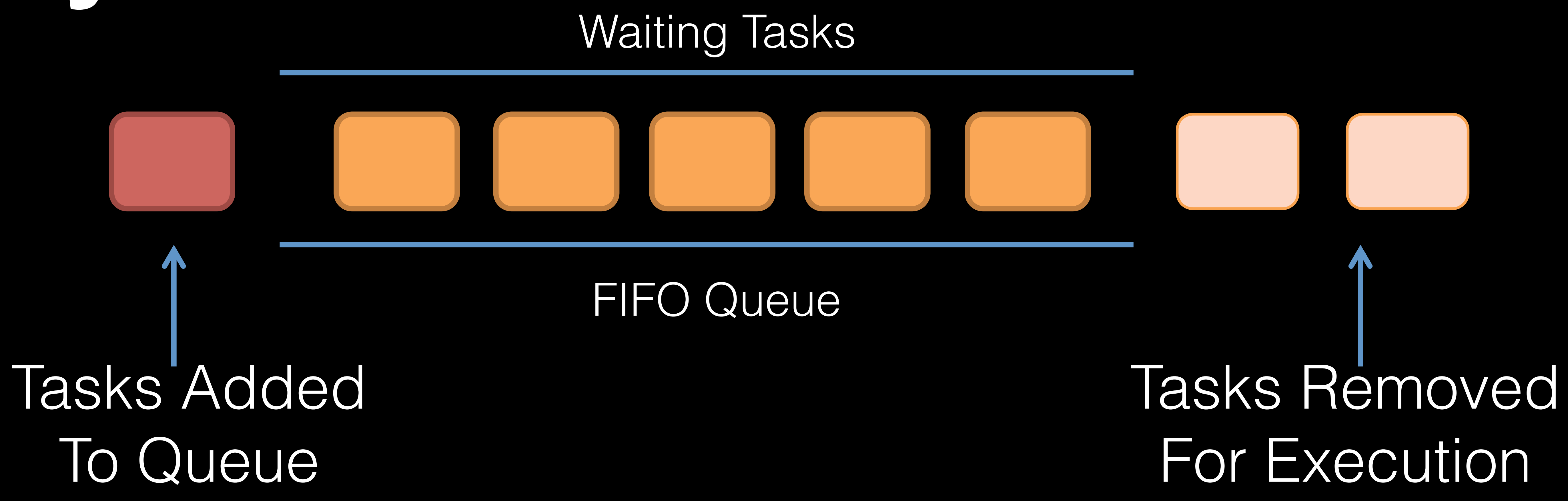
Classes

```
class Square {

    var area: Double = 0

    init(sideLength: Double) {
        self.sideLength = sideLength
    }

    var sideLength: Double {
        get {
            return sqrt(area)
        }
        set {
            area = newValue * newValue
        }
    }
}
```

Protocols and Extensions

```
protocol Describable {
    func description() -> String
}

struct Car : Describable {
    func description() -> String {
        return "Goes vroom”
    }
}

extension Double : Describable {
    func description() -> String {
        return "Currently set to \(self)”
    }
}

let d: Double = 3
print(d.description()) // “Currently set to 3.0”
```

Concurrency



```
import Dispatch

let serialQueue = DispatchQueue(label: "serial queue")

serialQueue async {
    print("run on serial queue")
}
serialQueue asyncAfter deadline: .now() + DispatchTimeInterval.seconds(1)) {
    print("run on serial queue")
}

let concurrentQueue = DispatchQueue(label: "concurrent queue", attributes: .concurrent)

concurrentQueue async {
    print("run on concurrent queue")
}
```

Swift @ IBM

Concurrency

```swift
import Dispatch

let queue = DispatchQueue(label: "group queue", attributes: .concurrent)
let group = DispatchGroup()

queue.async(group: group) {
    print("work 1")
}

queue.async(group: group) {
    print("work 2")
}

group.notify(queue: queue) {
    print("all work completed")
}
```

Swift Programming

Safe | Expressive | Fast

Safe: Optionals

Java

```
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}
```

Safe: Optionals

Java

```java
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}
```

Swift

```swift
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)
```

Safe: Optionals

Java

```
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}

> javac Test.java
```

Swift

```
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)




> swiftc main.swift
```

Safe: Optionals

Java

```java
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}
```

```
> javac Test.java
>
```

Swift

```swift
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)
```

```
> swiftc main.swift
```

Safe: Optionals

Java

```
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}

> javac Test.java
>
```

Swift

```
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)




> swiftc main.swift
> Error line 7: Value of optional type ‘String?’ not 
unwrapped;
> did you mean to use ‘!’ or ‘?’?
```

Safe: Optionals

Java

```java
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}

> javac Test.java
>
> java Test
```

Swift

```swift
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)



> swiftc main.swift
> Error line 7: Value of optional type ‘String?’ not
unwrapped;
> did you mean to use ‘!’ or ‘?’?
```

Safe: Optionals

Java

```
public class Test {

    private static void length (String string){
        System.out.println(string.length());
    }

    public static void main(String[] args){
        String string = null;

    length(string);
    }

}

> javac Test.java

> java Test

Exception in thread "main"
java.lang.NullPointerException
at Test.length(Test.java:5)
at Test.main(Test.java:11)
```

Swift

```
func length(of string: String) -> Void {
    print(string.characters.count)
}


var str: String? = nil

length(of: str)




> swiftc main.swift
> Error line 7: Value of optional type ‘String?’ not
unwrapped;
> did you mean to use ‘!’ or ‘?’?
```

Swift Programming

Safe | **Expressive** | Fast

Expressive: Functions

```
func addInts(a: Int, b: Int) -> Int {

}
```

Expressive: Functions

```swift
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
```

Expressive: Functions

```
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)
```

Expressive: Functions

```swift
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)

func addInts(_ a: Int, _ b: Int) -> Int {
    return a + b
}
```

Expressive: Functions

```
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)

func addInts(_ a: Int, _ b: Int) -> Int {
    return a + b
}
addInts(1, 3)
```

Expressive: Functions

```
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)

func addInts(_ a: Int, _ b: Int) -> Int {
    return a + b
}
addInts(1, 3)


func move(from start: Point, to end: Point) -> Bool { /* code */ }
```

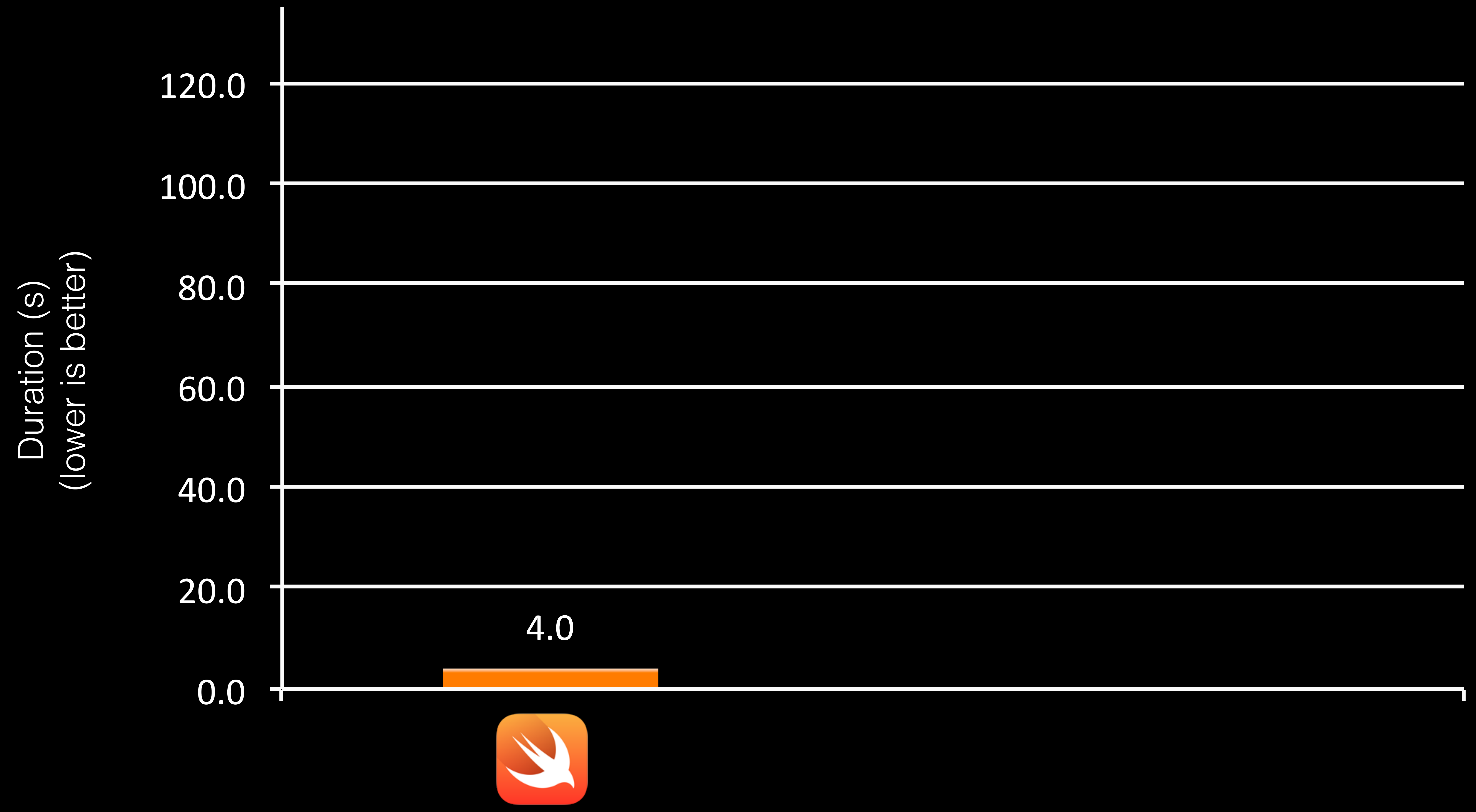
Expressive: Functions

```swift
func addInts(a: Int, b: Int) -> Int {
    return a + b
}
addInts(a: 1, b: 3)

func addInts(_ a: Int, _ b: Int) -> Int {
    return a + b
}
addInts(1, 3)


func move(from start: Point, to end: Point) -> Bool { /* code */ }

move(from: a, to: b)
```

Swift Programming

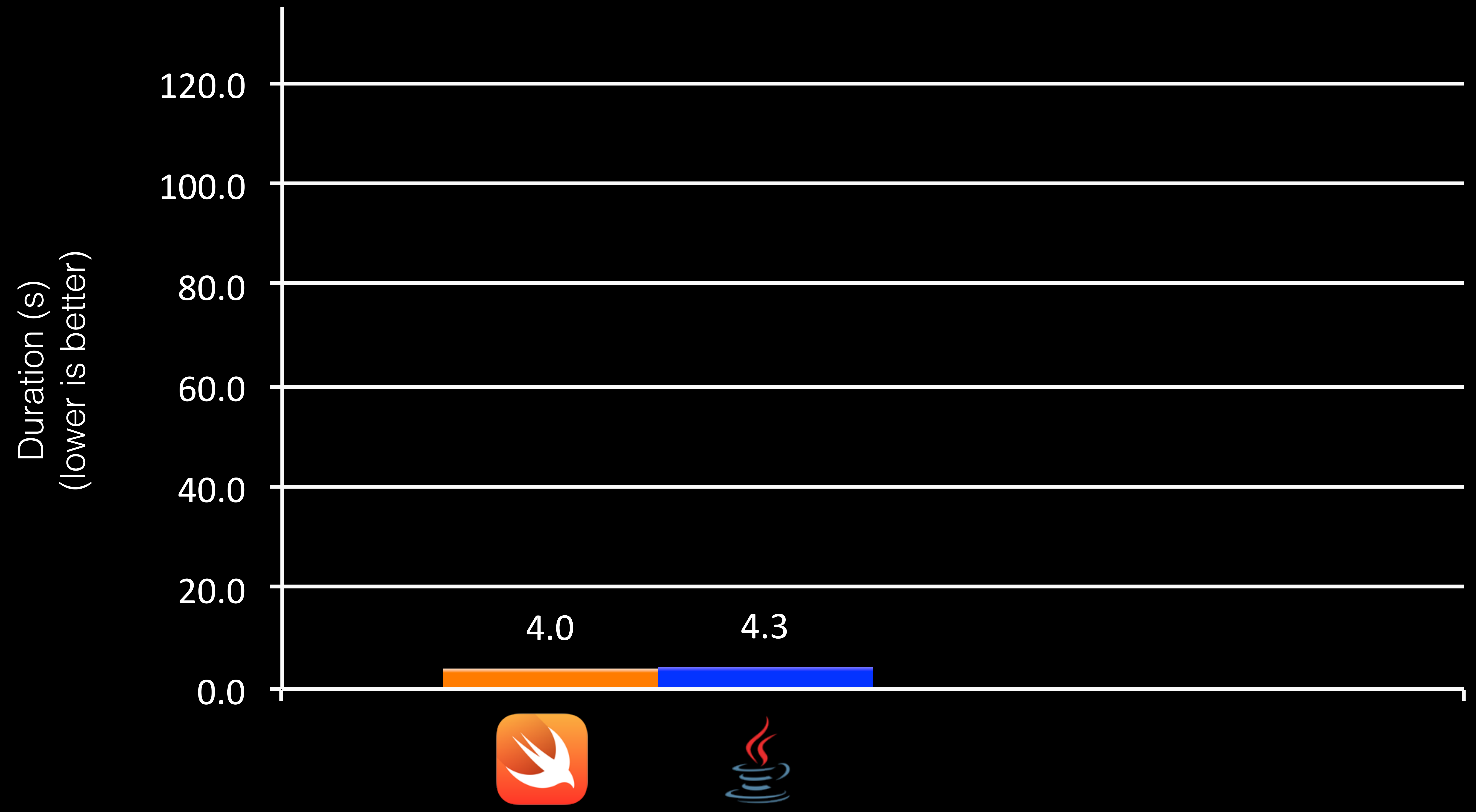
Safe | Expressive | **Fast**

Fast: Performant Applications



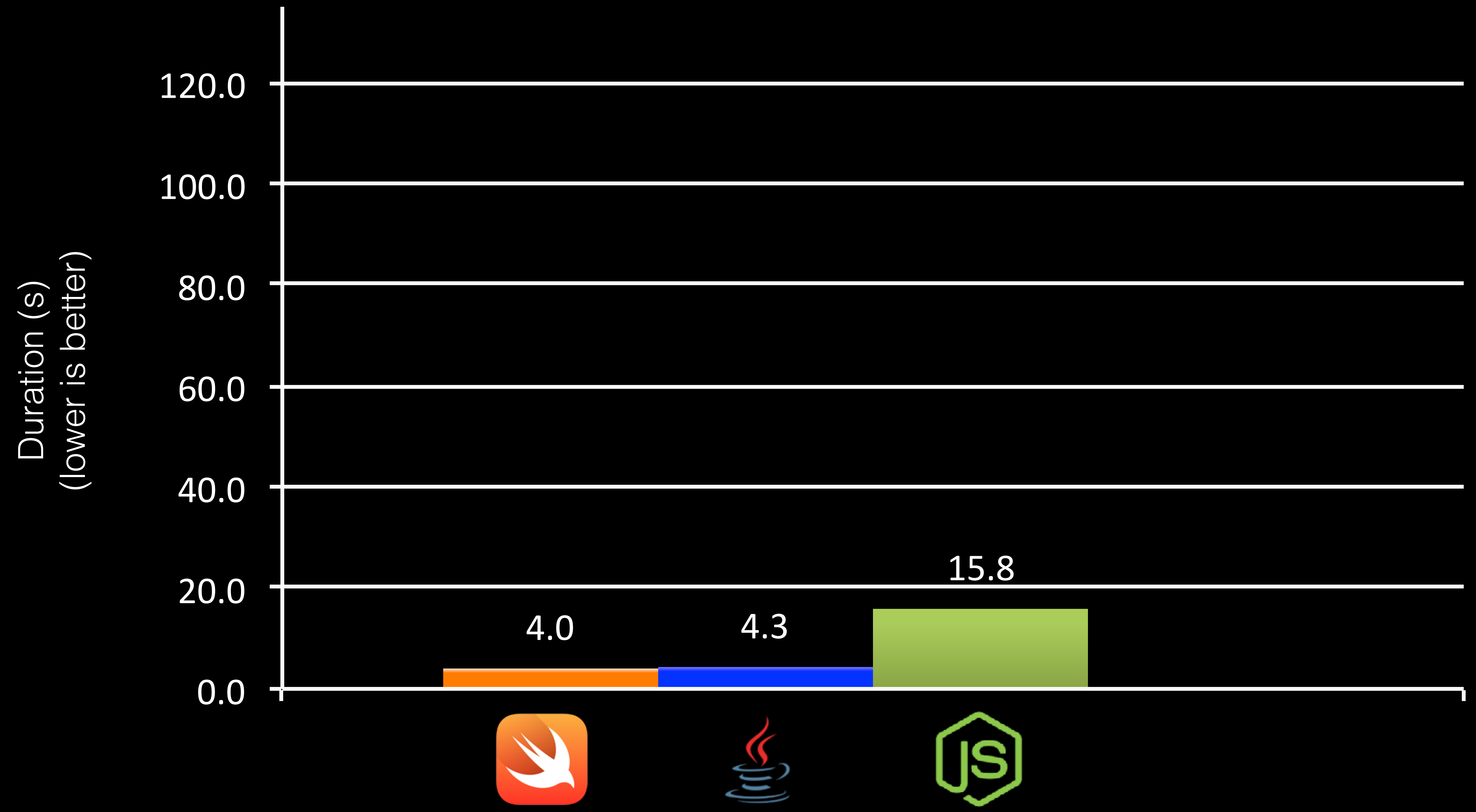
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Fast: Performant Applications



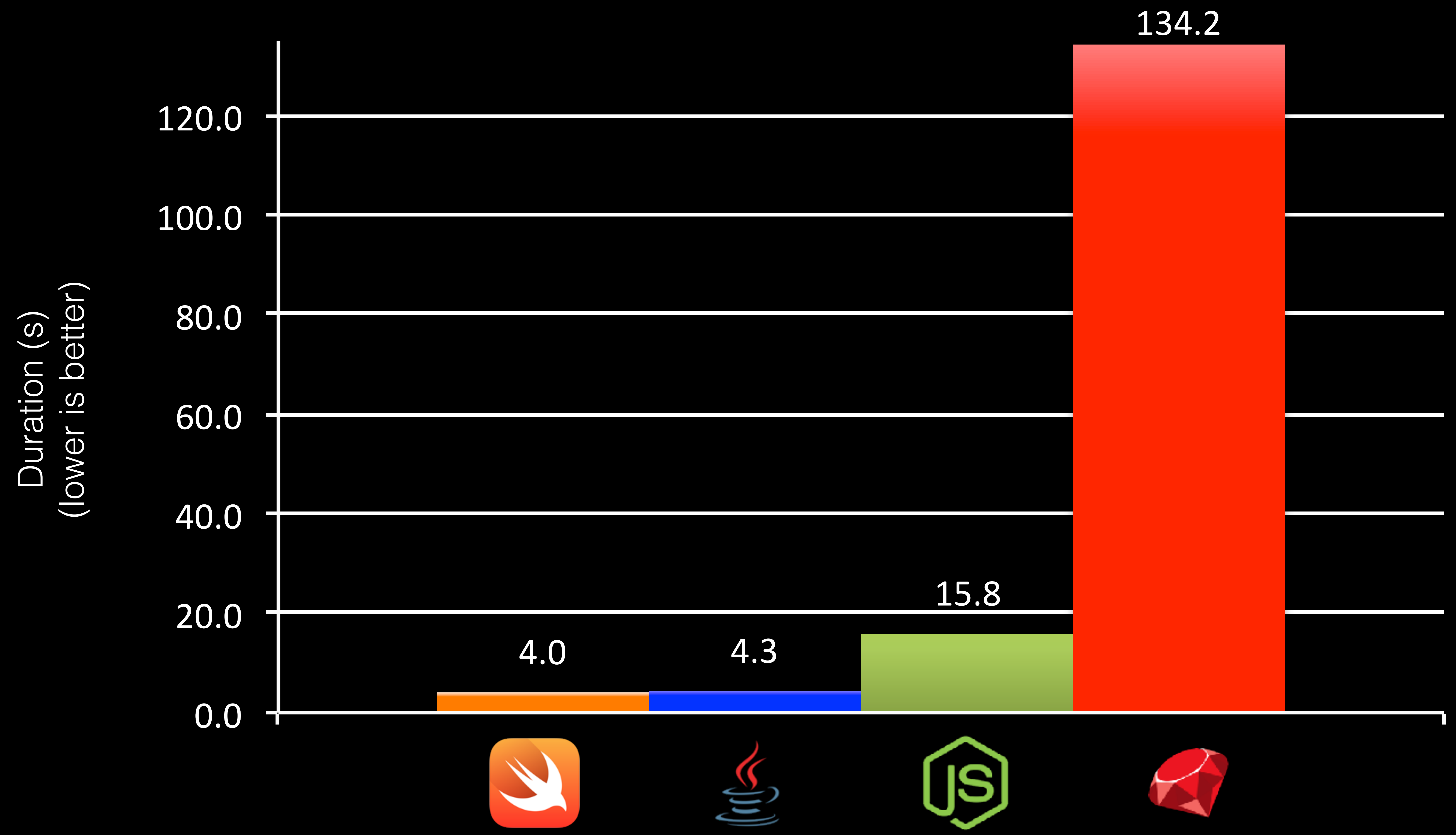
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Fast: Performant Applications



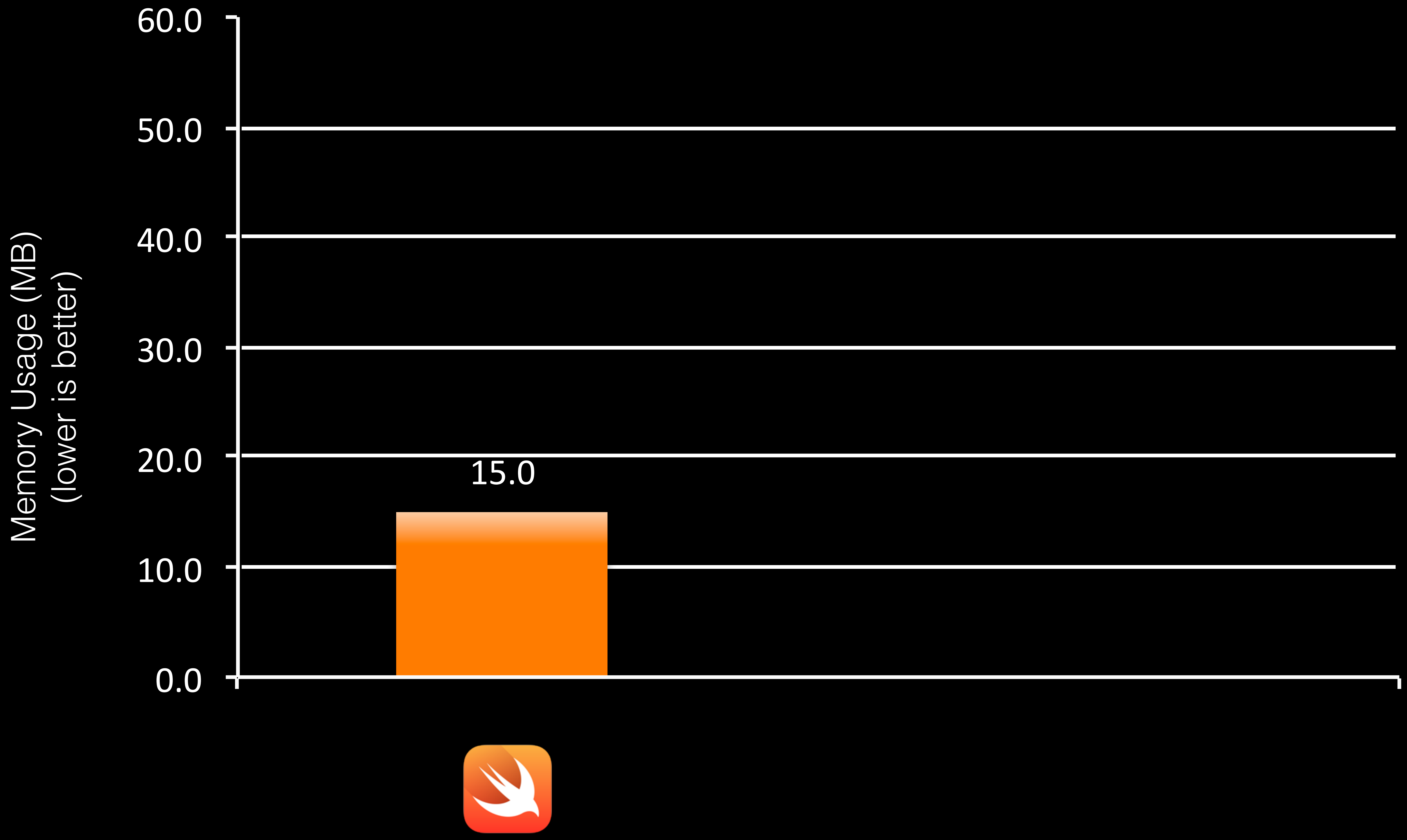
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Fast: Performant Applications



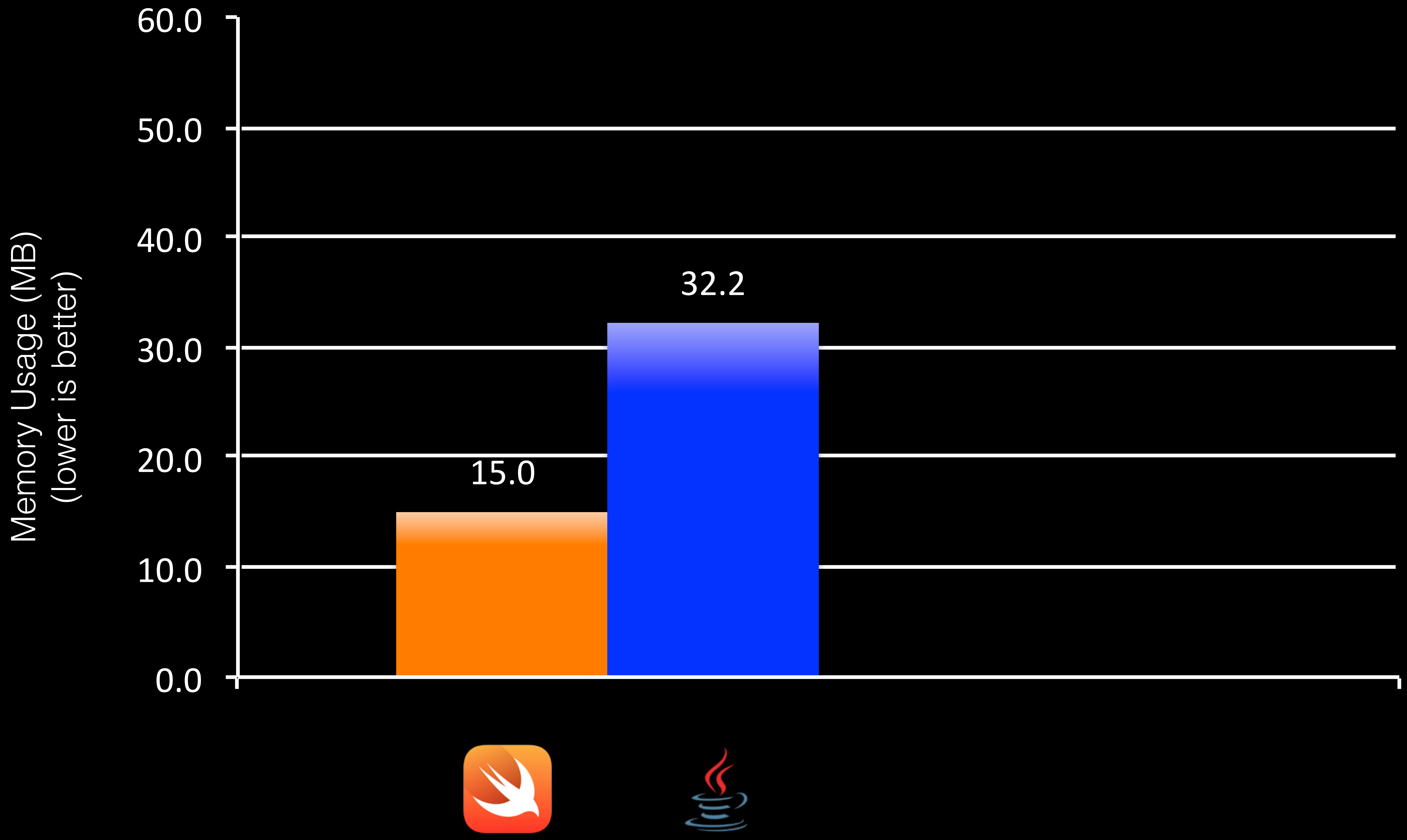
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Low Memory



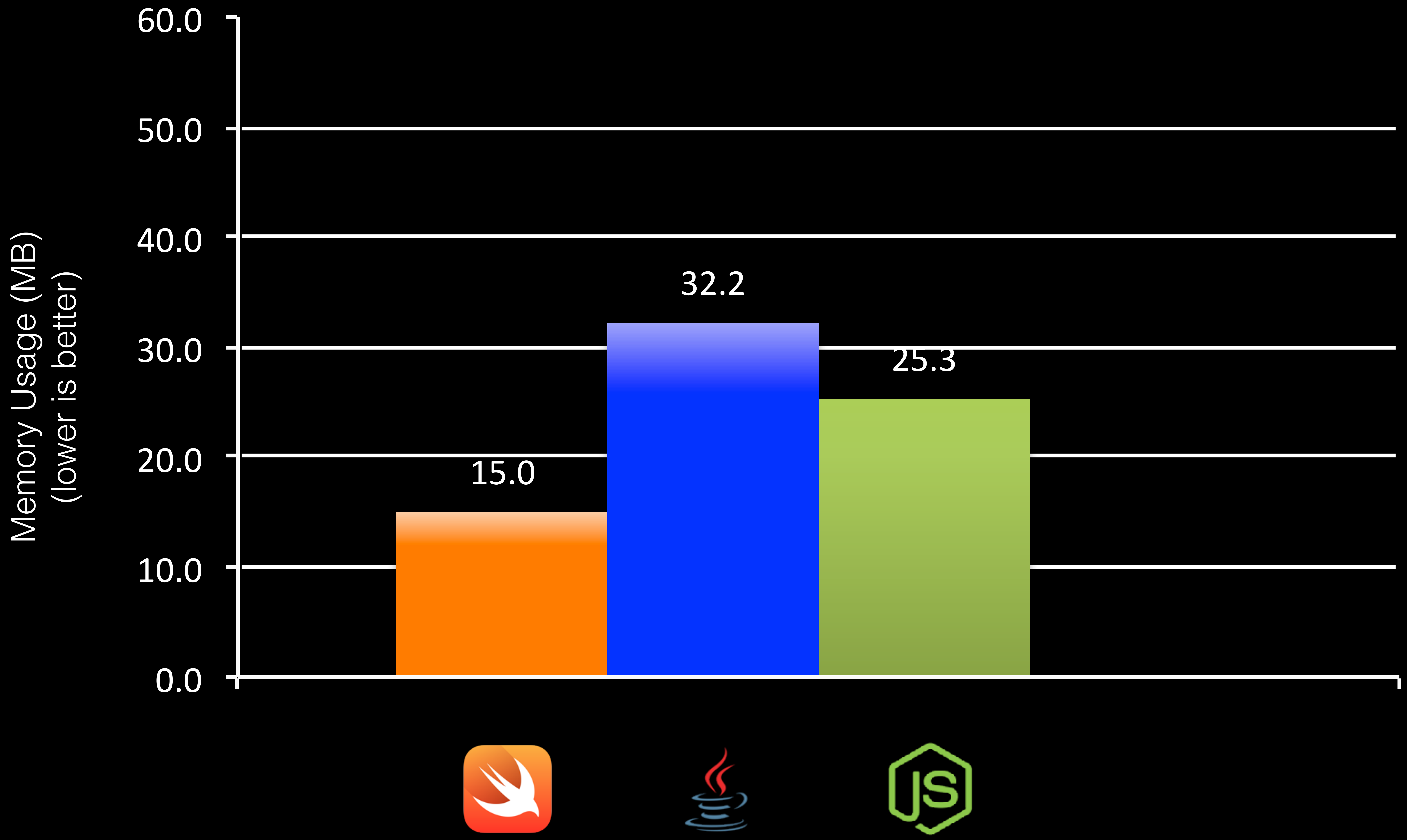
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Low Memory



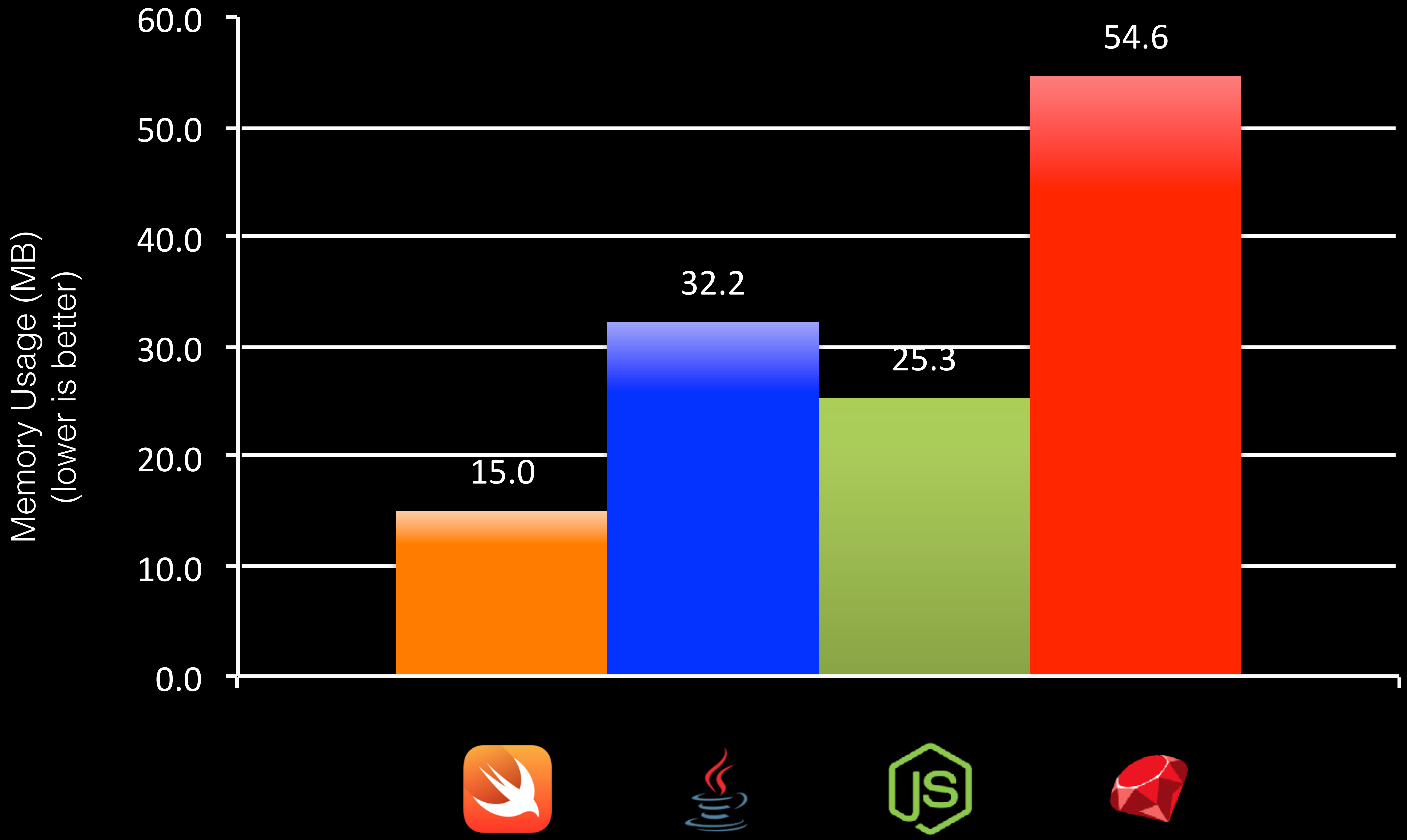
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Low Memory



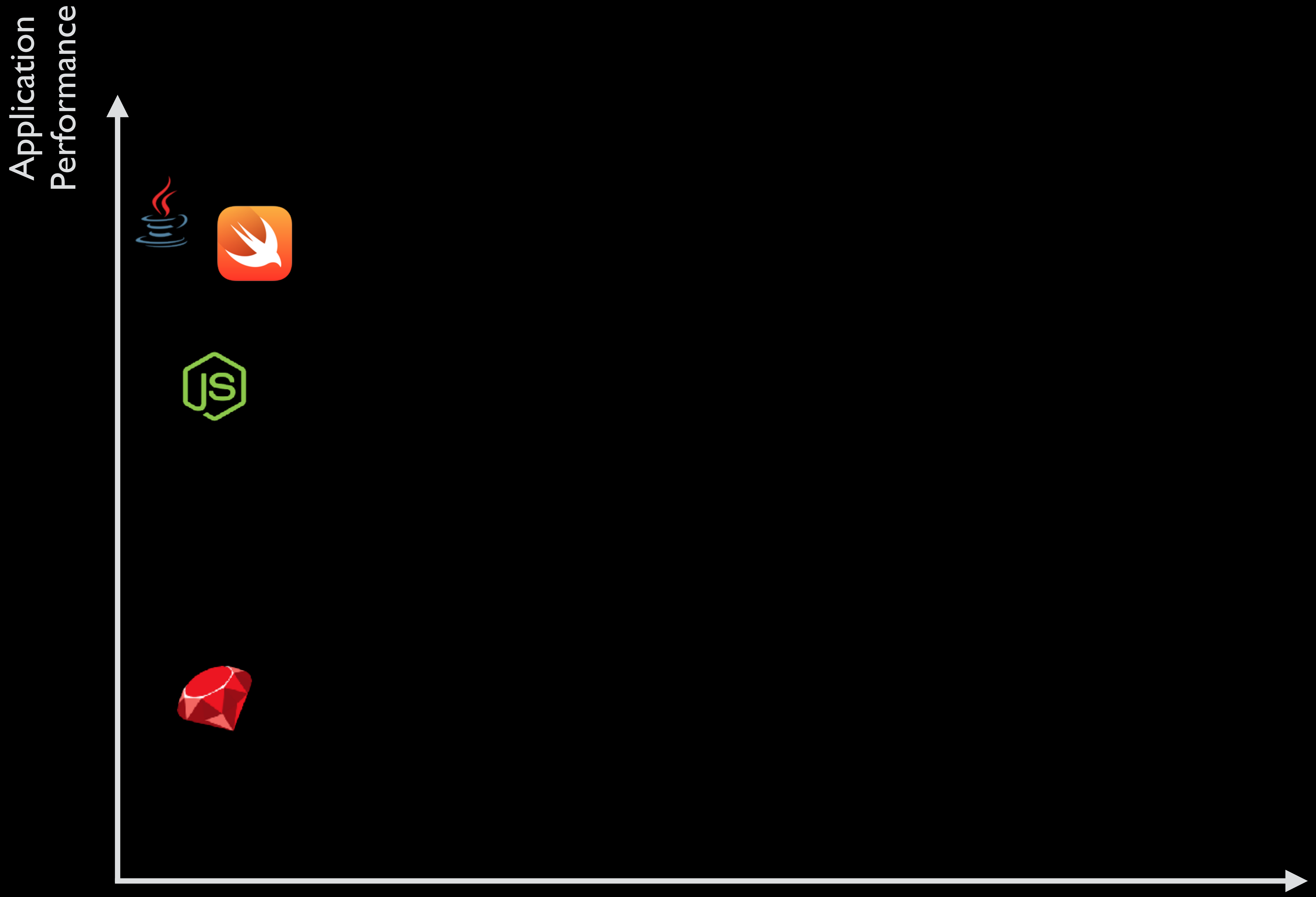
http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Low Memory

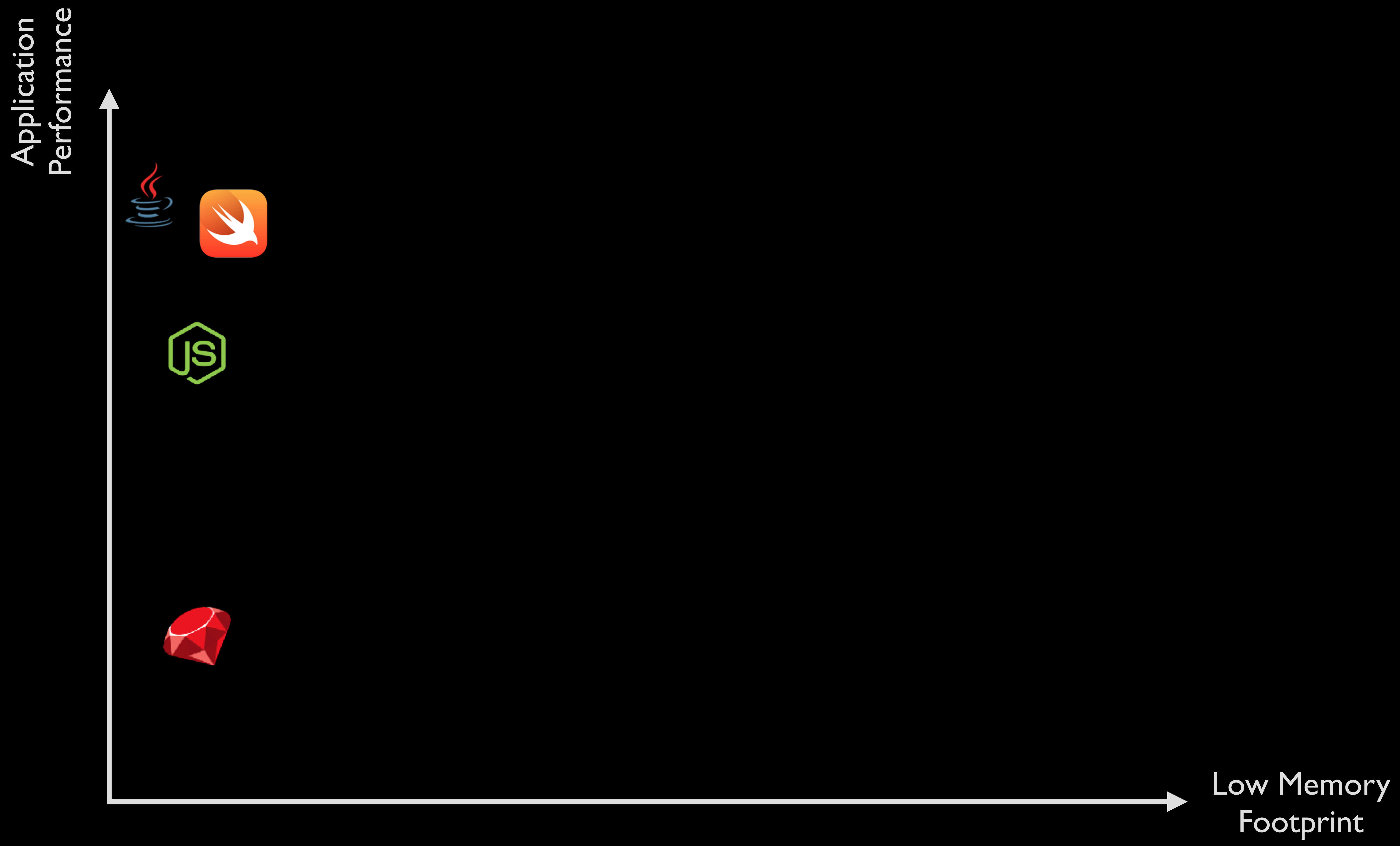


http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Programming Languages



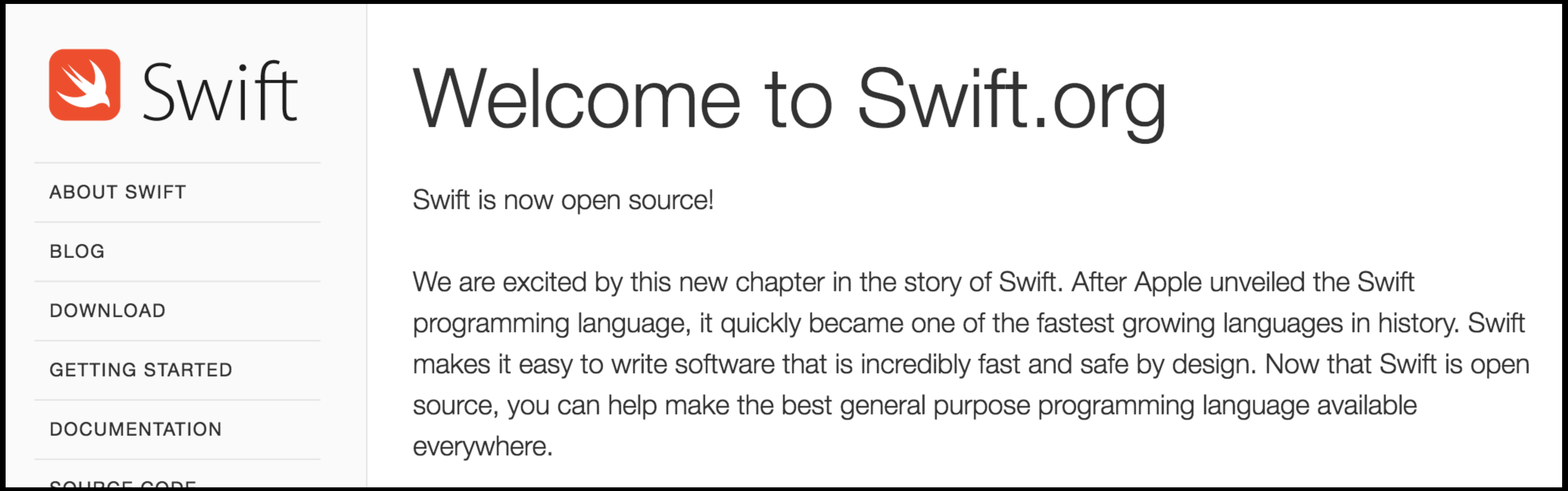
Programming Languages



Swift on the Server

Progress of Swift

December 3rd, 2015



Welcome to Swift.org

Swift is now open source!

We are excited by this new chapter in the story of Swift. After Apple unveiled the Swift programming language, it quickly became one of the fastest growing languages in history. Swift makes it easy to write software that is incredibly fast and safe by design. Now that Swift is open source, you can help make the best general purpose programming language available everywhere.

THE APACHE® SOFTWARE FOUNDATION

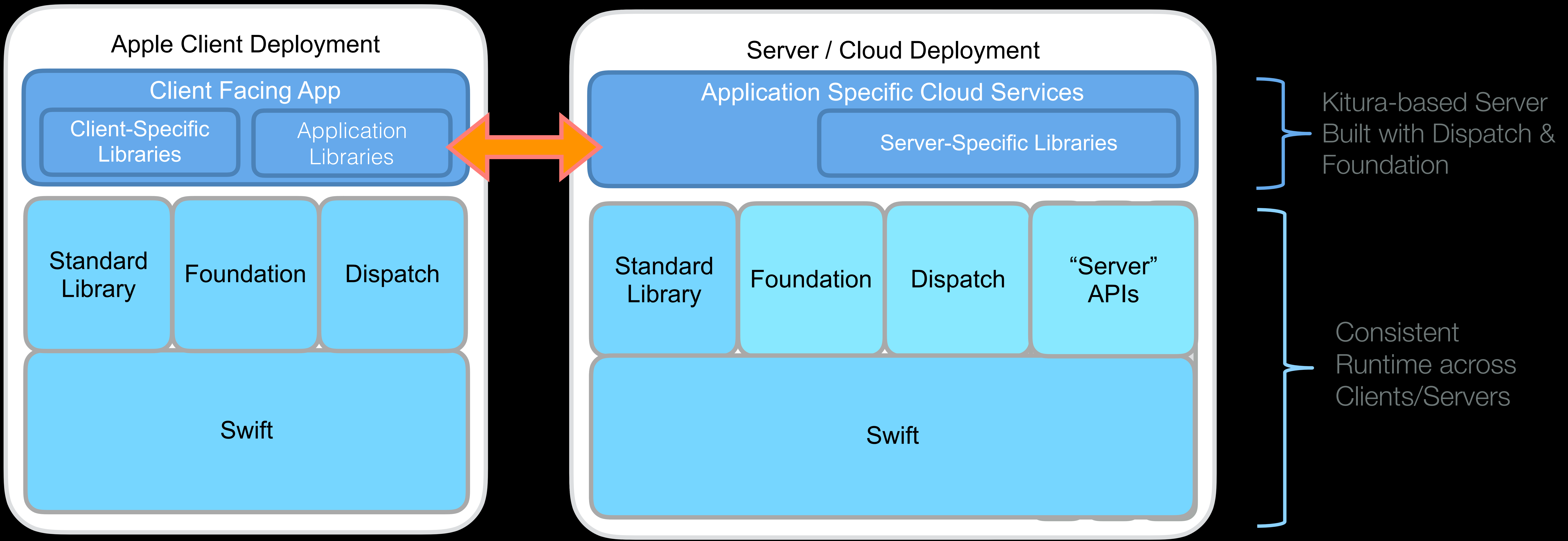
Apache 2.0 Software Licence

IBM LinuxONE™

Kitura: A Swift Web Framework and HTTP Server

http://kitura.io

Kitura Web Framework


Apple Client Deployment
Client Facing App
Client-Specific Libraries
Application Libraries
Standard Library
Foundation
Dispatch
Swift
Server / Cloud Deployment
Application Specific Cloud Services
Server-Specific Libraries
Standard Library
Foundation
Dispatch
“Server” APIs
Swift
Kitura-based Server Built with Dispatch & Foundation
Consistent Runtime across Clients/Servers

Swift 3.0 + Kitura 1.0
Swift on the Server is Real

Lets Take a Tour

Create an Application

First, create a new project directory:

```
$ mkdir myFirstProject
```

Next, create a new Swift project using the Swift Package Manager.

```
$ cd myFirstProject
$ swift package init --type executable
```

In **`Package.swift`**, add Kitura as a dependency for your project.

```swift
import PackageDescription

let package = Package(
    name: "myFirstProject",
    dependencies: [
        .Package(url: "https://github.com/IBM-Swift/Kitura.git", majorVersion: 1, minor: 0)
    ])
```

Create an Application

In **`Sources/main.swift`**, add the following code.

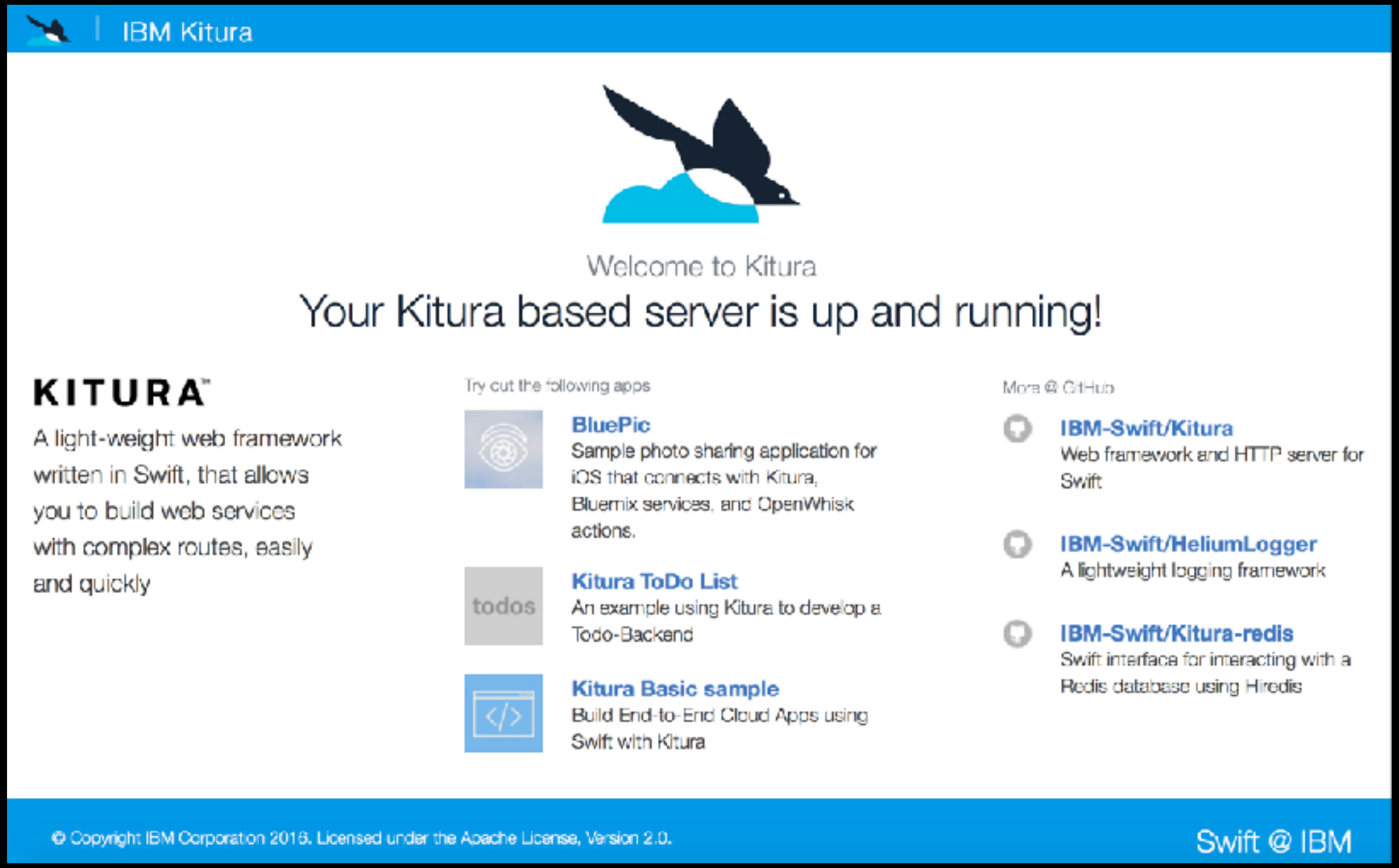
```swift
import Kitura

// Create a new router
let router = Router()

// Handle HTTP GET requests to /
router.get("/") {
    request, response, next in
    response.send("Hello, World!")
    next()
}

// Add an HTTP server and connect it to the router
Kitura.addHTTPServer(onPort: 8090, with: router)

// Start the Kitura runloop (this call never returns)
Kitura.run()
```

Run an Application

Compile and run your application:

```
$ swift build
$ .build/debug/myFirstProject
```

Open your browser at http://localhost:8090



IBM Kitura

Welcome to Kitura

Your Kitura based server is up and running!

KITURA™

A light-weight web framework written in Swift, that allows you to build web services with complex routes, easily and quickly

Try out the following apps

BluePic
Sample photo sharing application for iOS that connects with Kitura, Bluemix services, and OpenWhisk actions.

Kitura ToDo List
An example using Kitura to develop a Todo-Backend

Kitura Basic sample
Build End-to-End Cloud Apps using Swift with Kitura

More @ GitHub

IBM-Swift/Kitura
Web framework and HTTP server for Swift

IBM-Swift/HeliumLogger
A lightweight logging framework

IBM-Swift/Kitura-redis
Swift interface for interacting with a Redis database using Hiredis

© Copyright IBM Corporation 2016. Licensed under the Apache License, Version 2.0.

Swift @ IBM

Demo

Use Services

MQTT.ORG

MySQL®

facebook

cassandra

Google

PostgreSQL

CouchDB
relax

GitHub

IBM Watson

redis

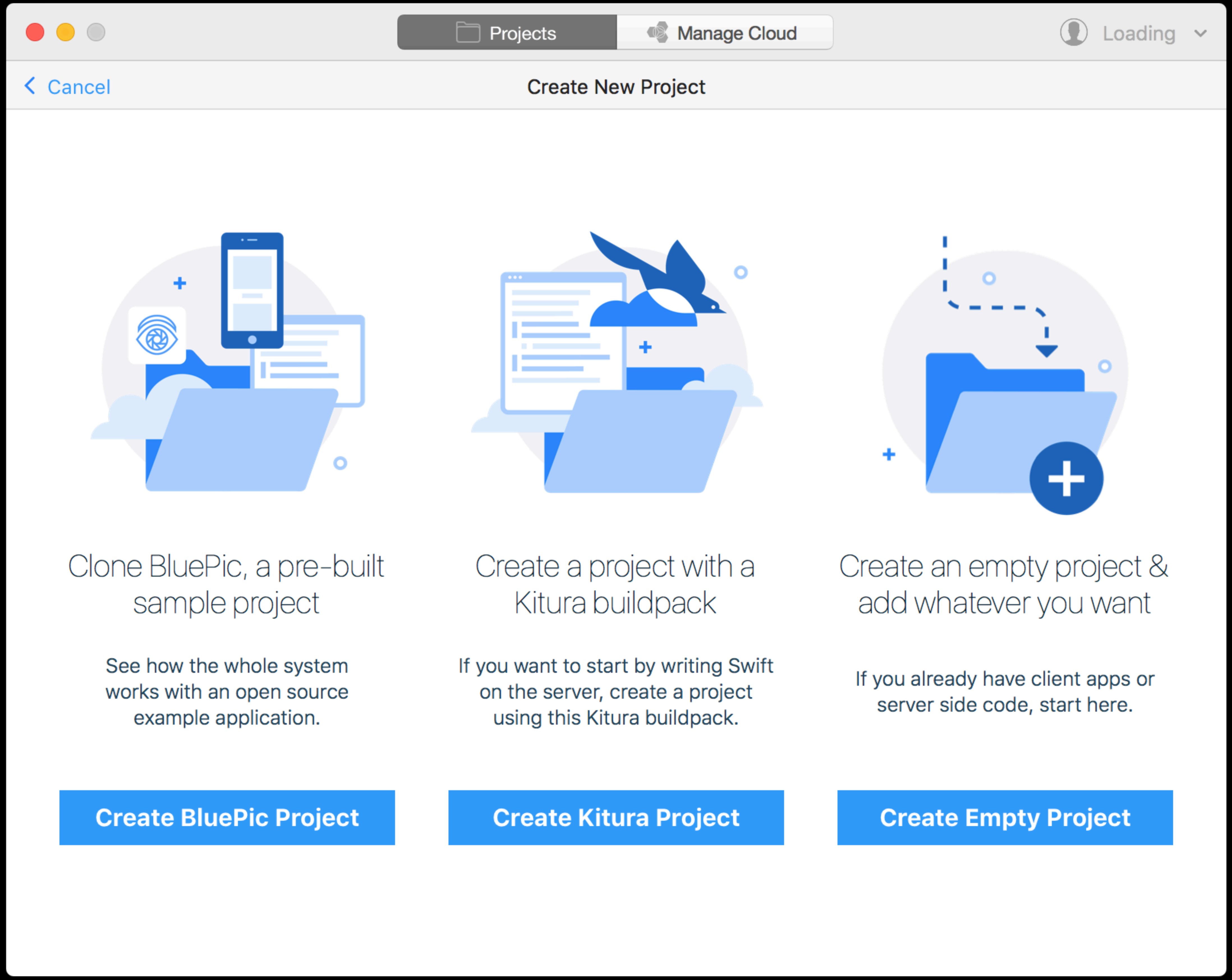
Deploy to Cloud

```
$ docker pull ibmcom/kitura-ubuntu:latest
```

```
$ git clone https://github.com/IBM-Swift/
Kitura-Starter-Bluemix
```

Using Cloud Tools

- Deployment made easy
- Clone, code, push
- Demo projects to try



Clone BluePic, a pre-built sample project

See how the whole system works with an open source example application.

Create a project with a Kitura buildpack

If you want to start by writing Swift on the server, create a project using this Kitura buildpack.

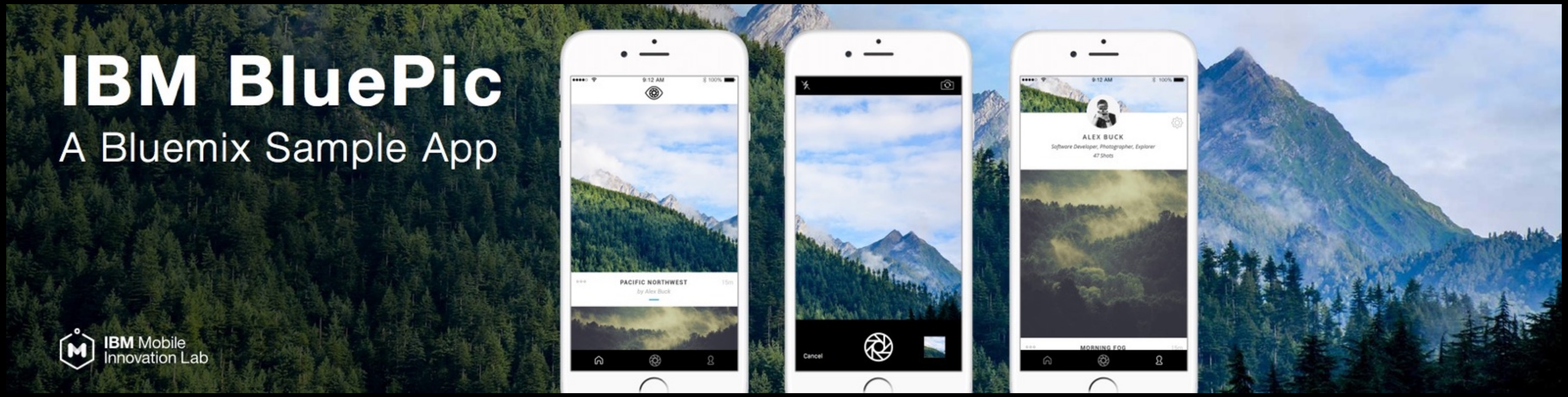
Create an empty project & add whatever you want

If you already have client apps or server side code, start here.

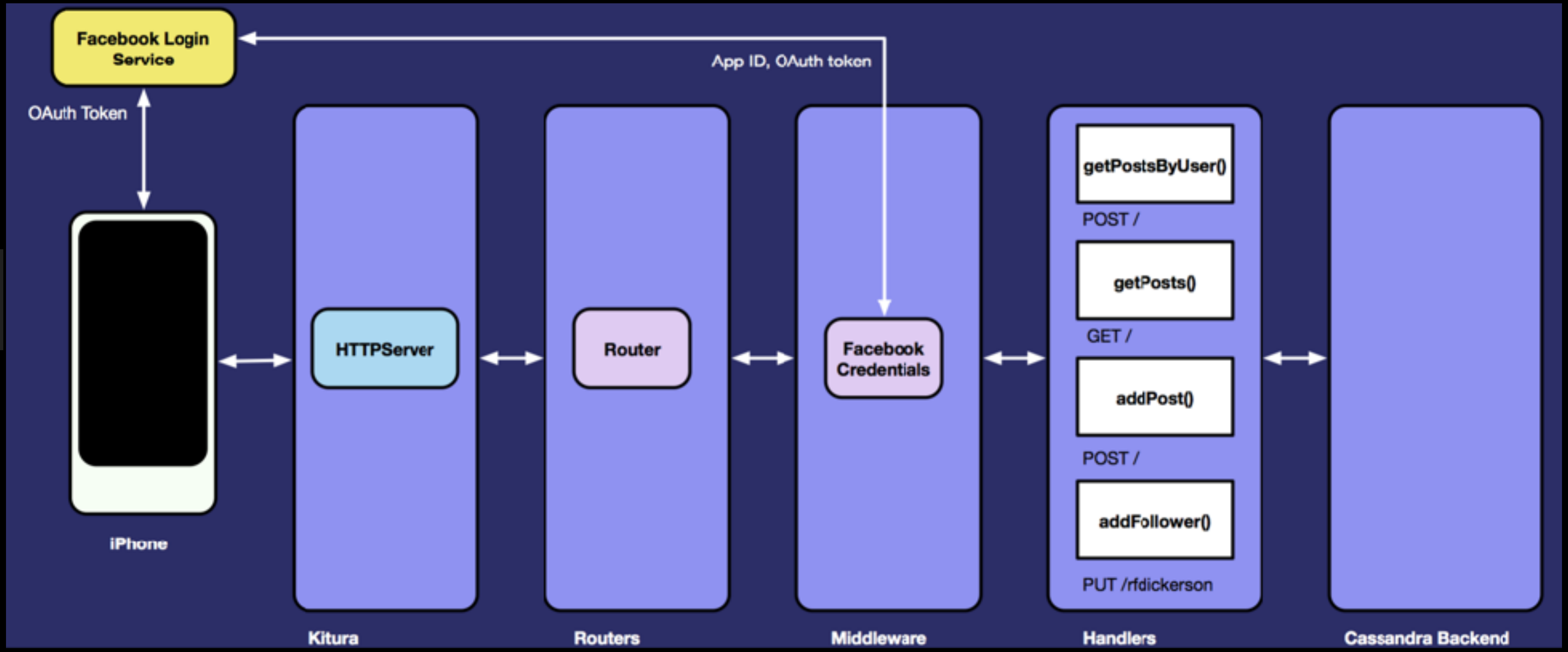
http://cloudtools.bluemix.net

Examples

BluePic Application
https://github.com/ibm-swift/bluepic

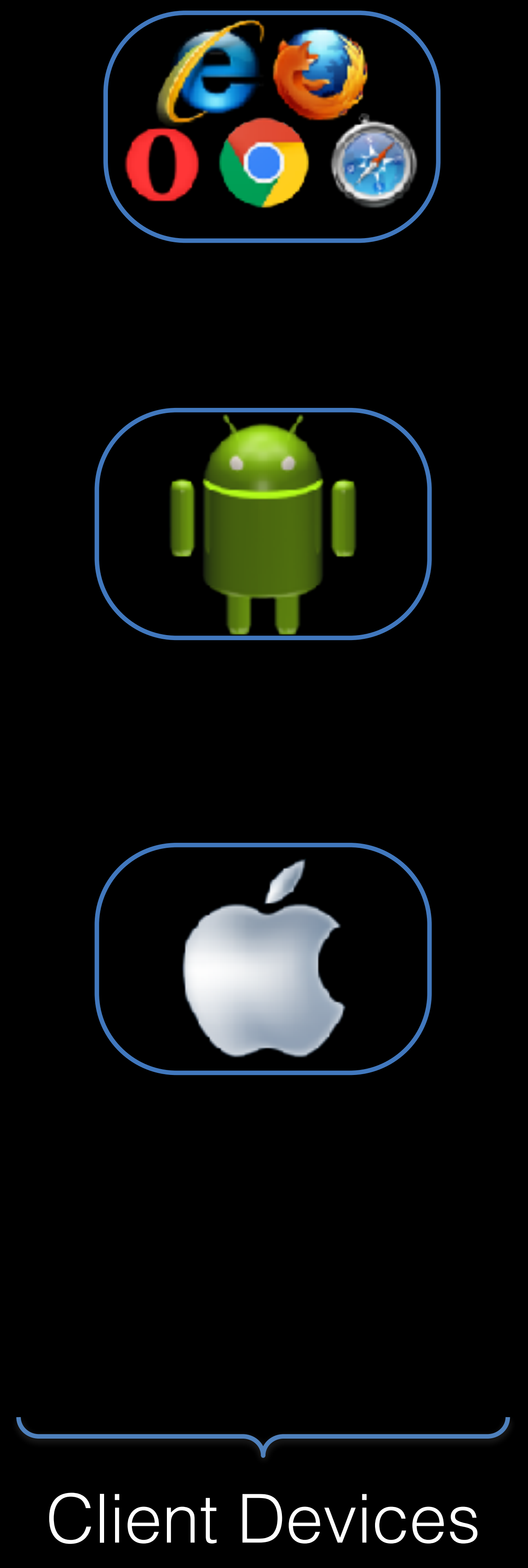


Blitter Social Network
https://github.com/ibm-swift/blitter

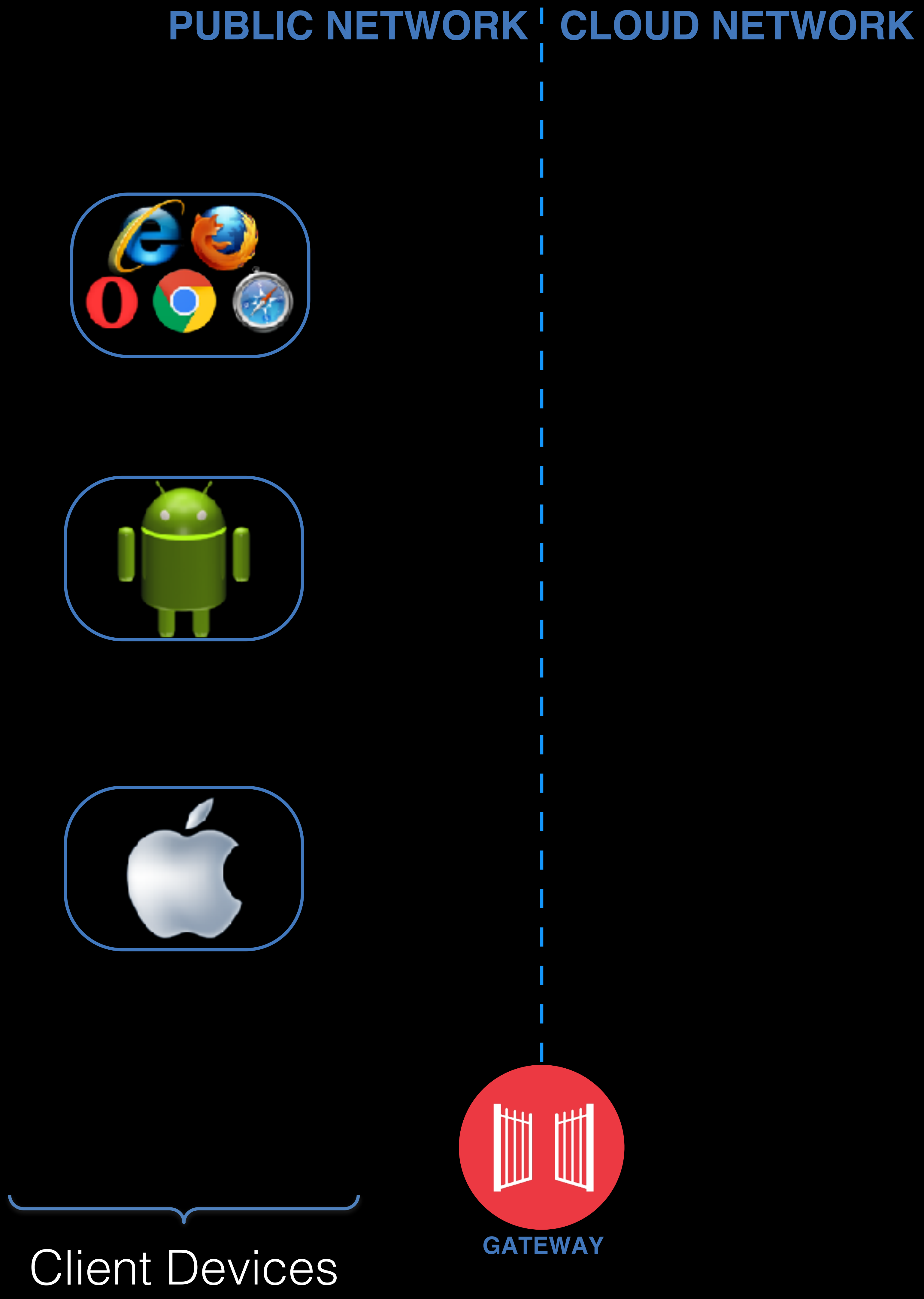


Usage Models

Recommended Swift Usage Model

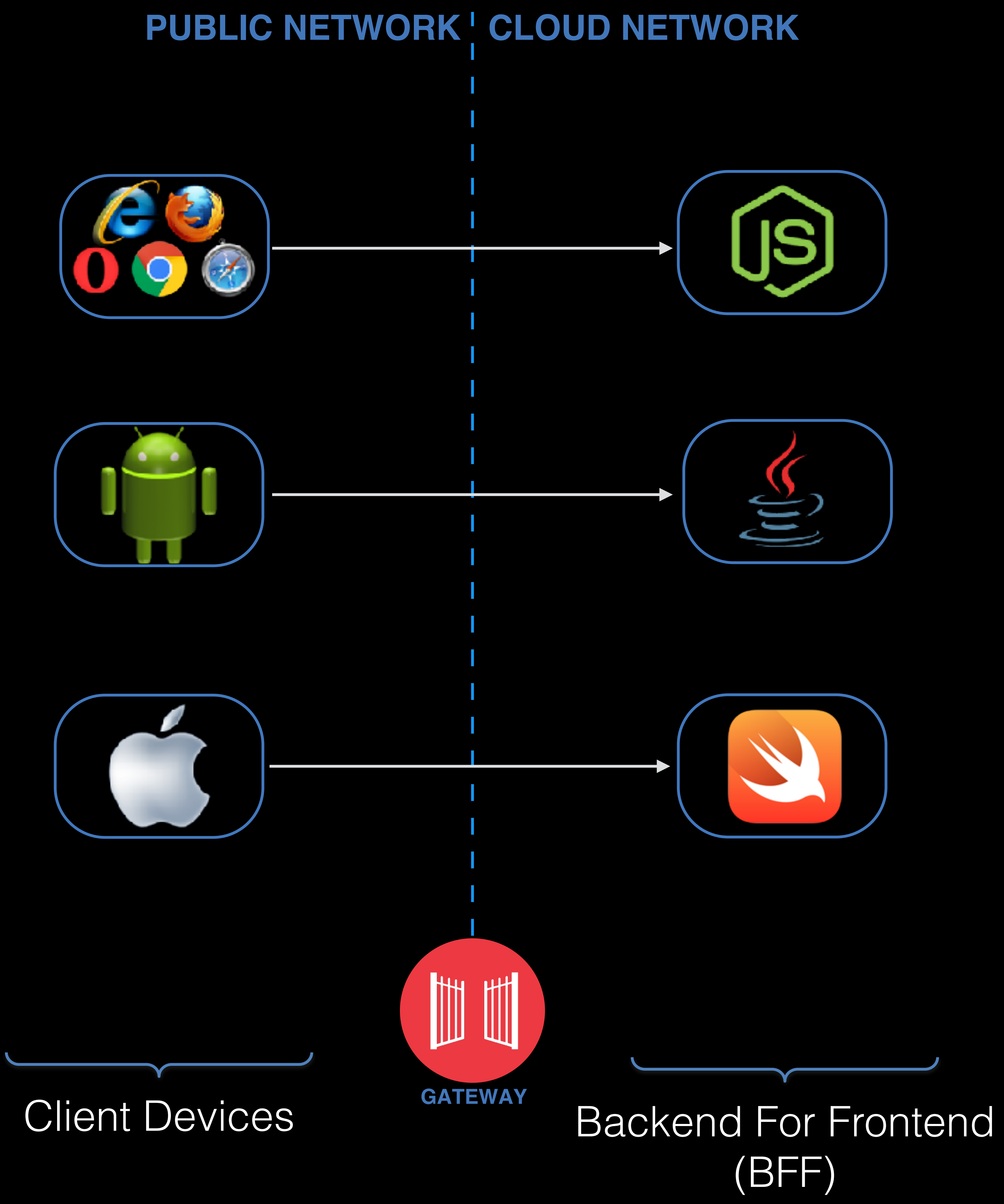


Recommended Swift Usage Model



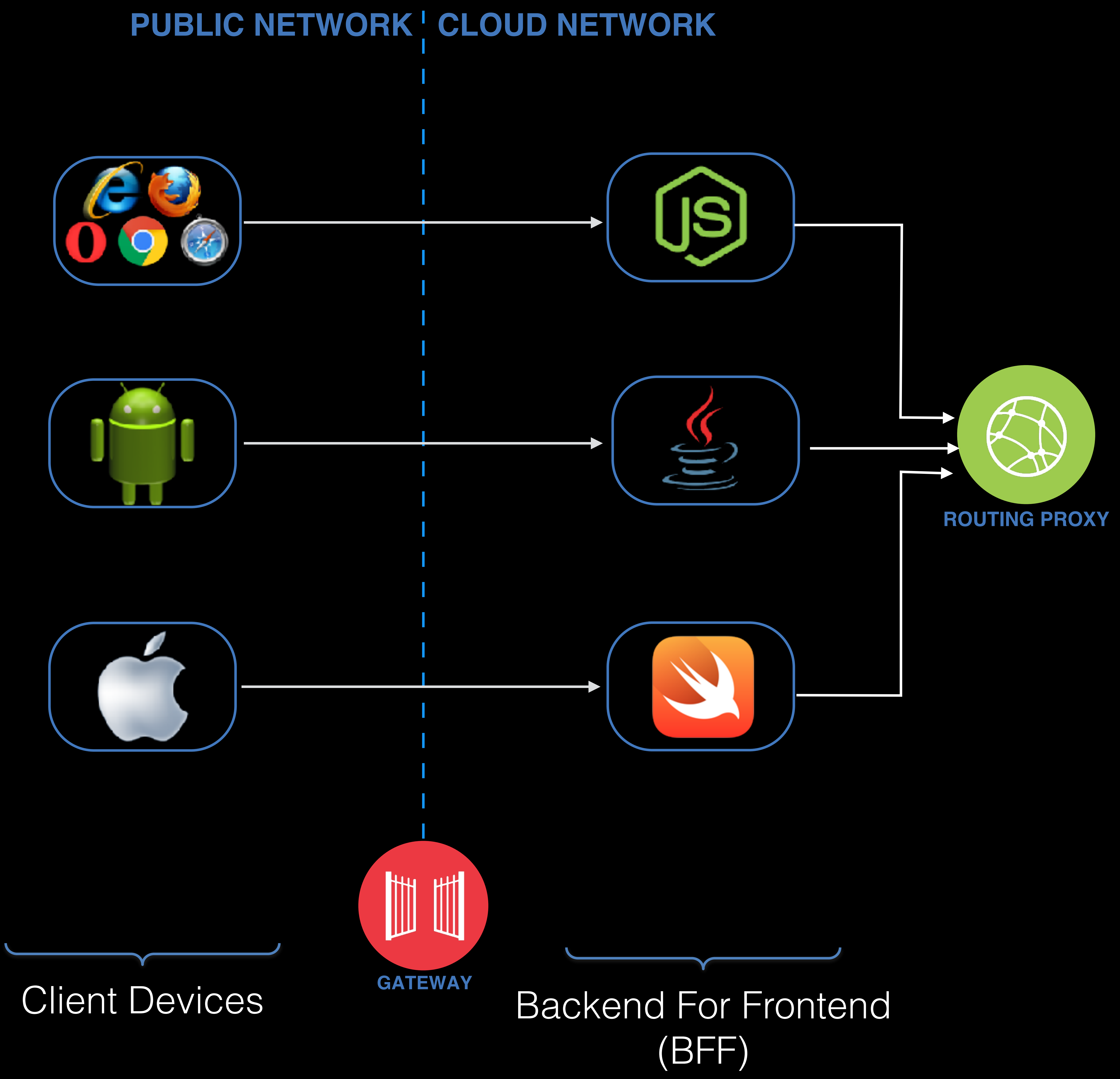
Swift @ IBM

Recommended Swift Usage Model

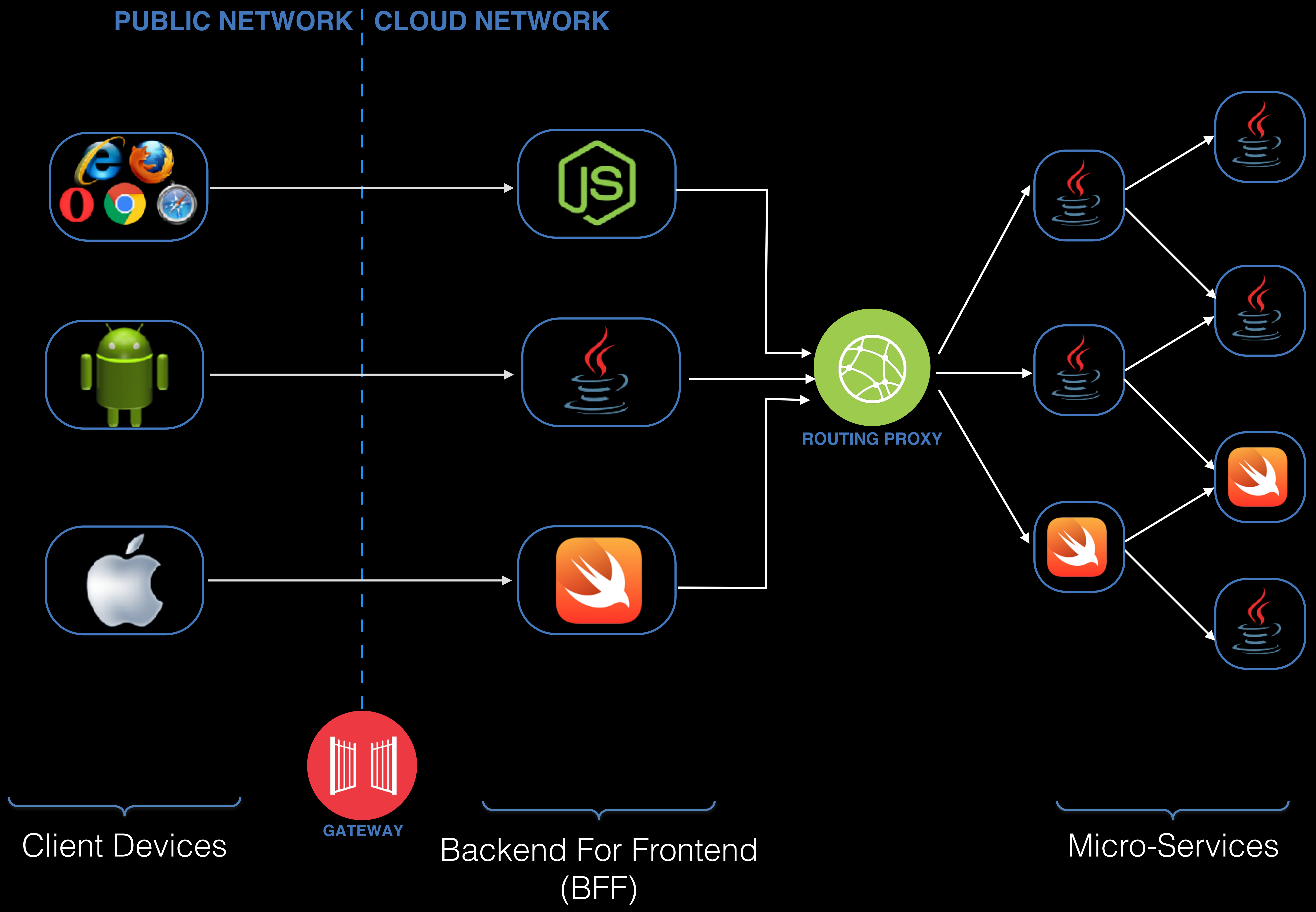


Swift @ IBM

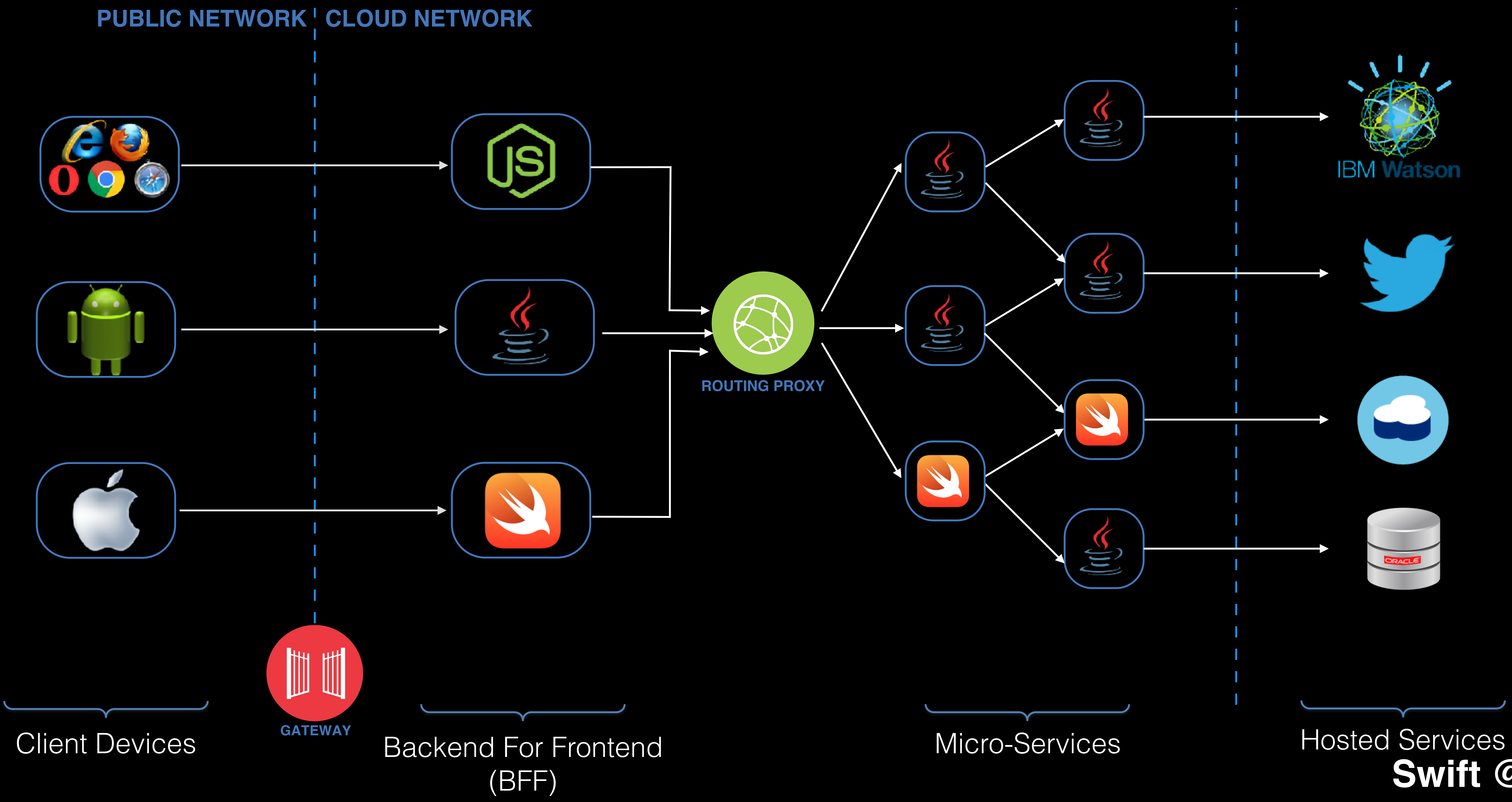
Recommended Swift Usage Model



Recommended Swift Usage Model

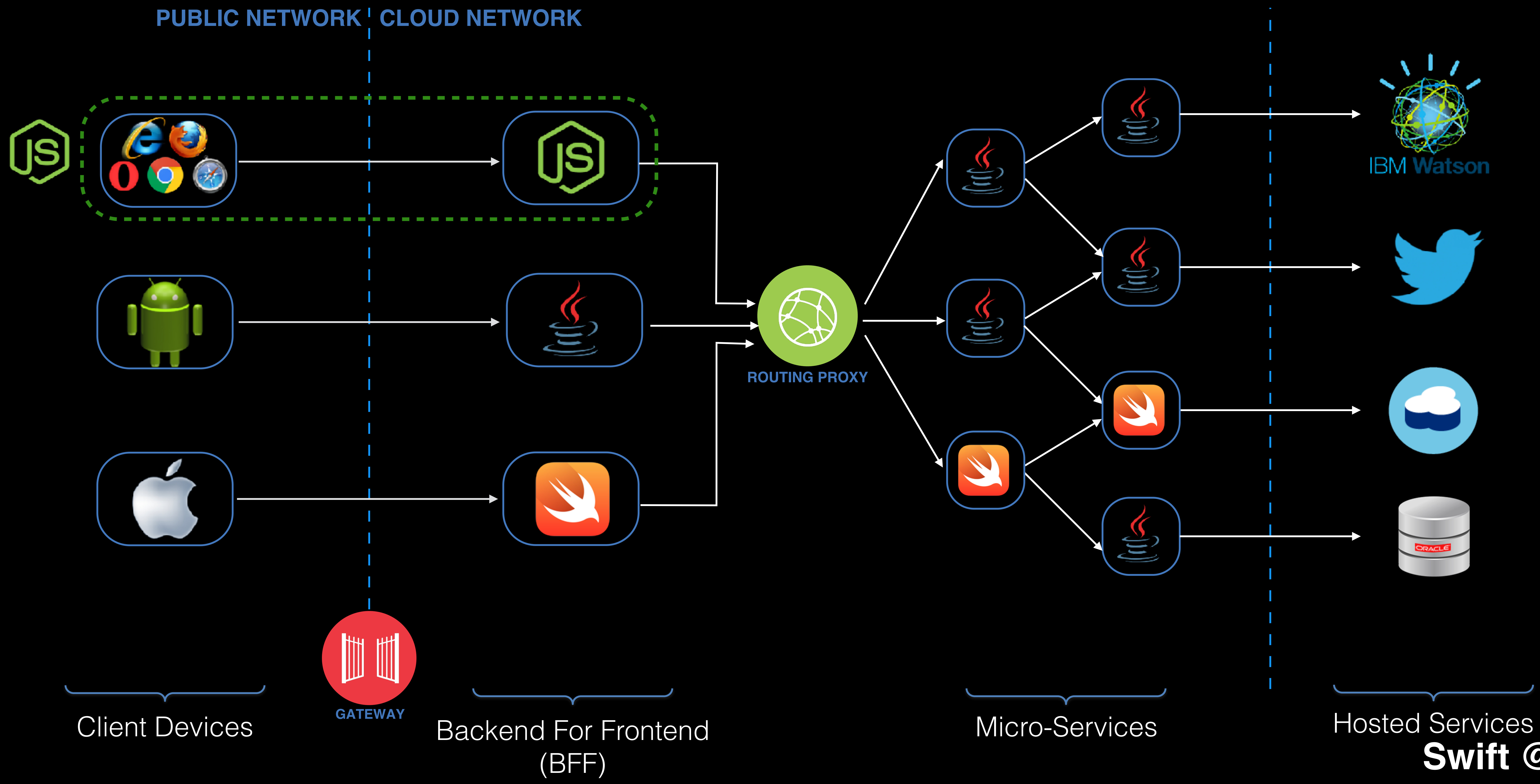


Recommended Swift Usage Model

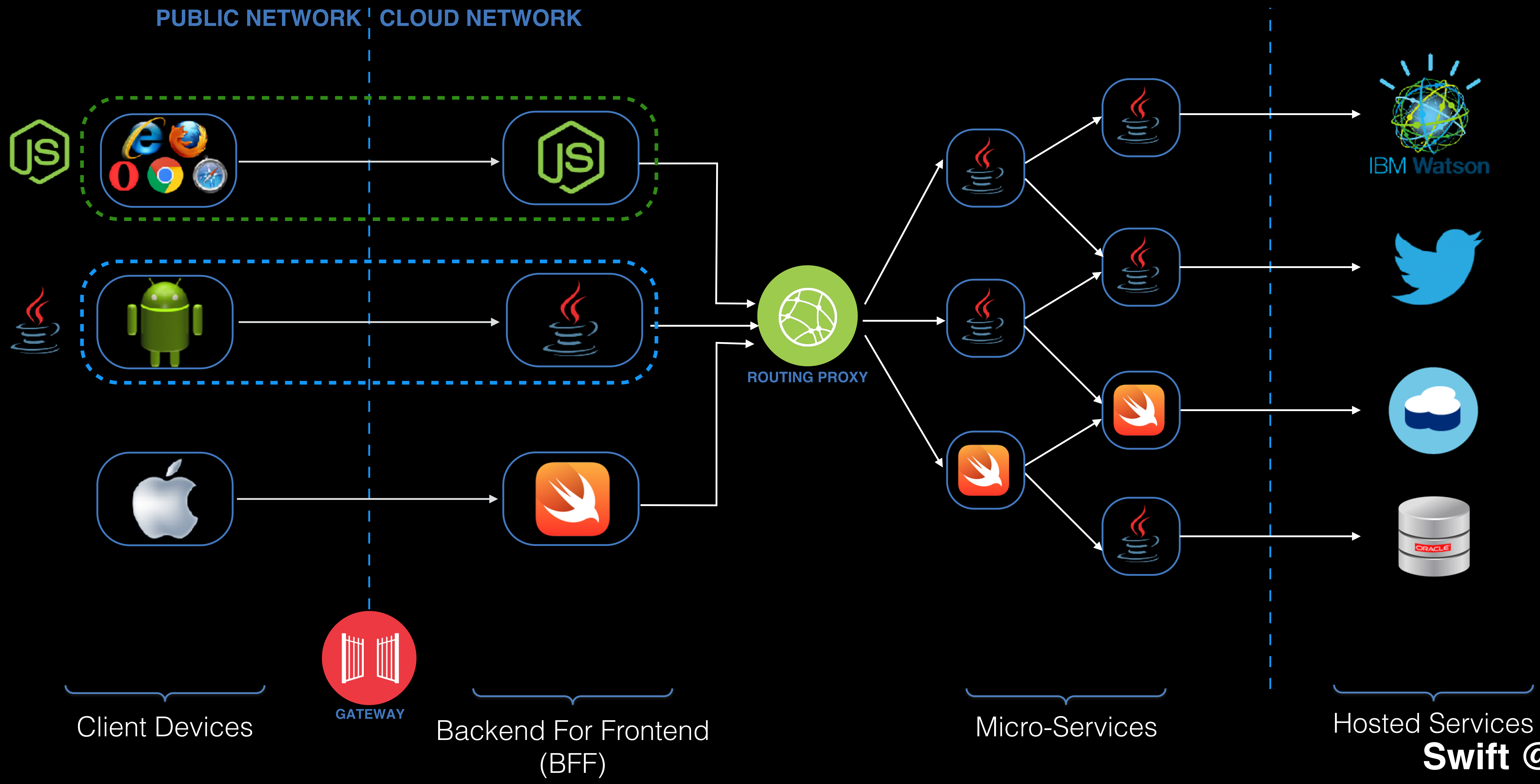


Swift @ IBM

Recommended Swift Usage Model

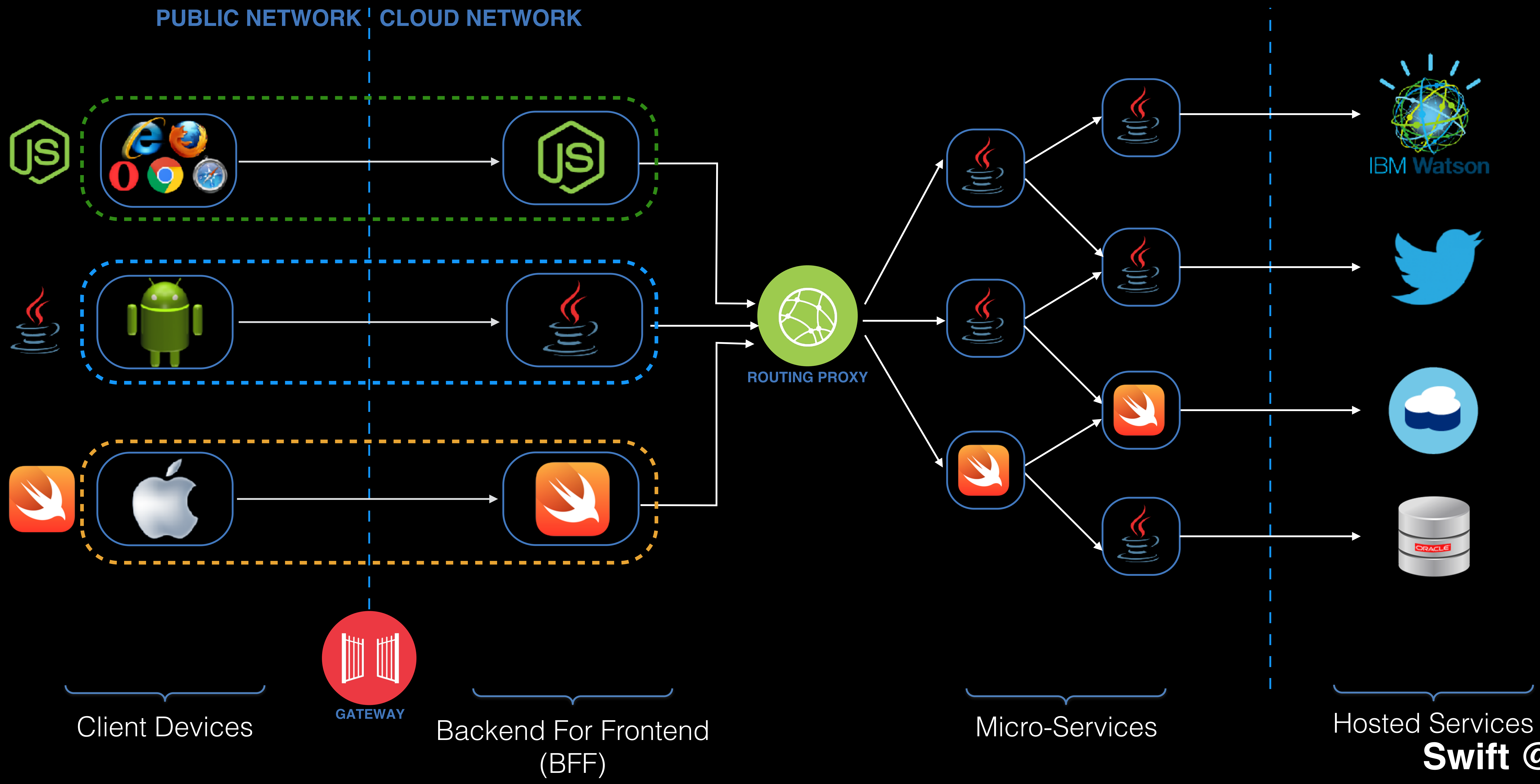


Recommended Swift Usage Model



Swift @ IBM

Recommended Swift Usage Model

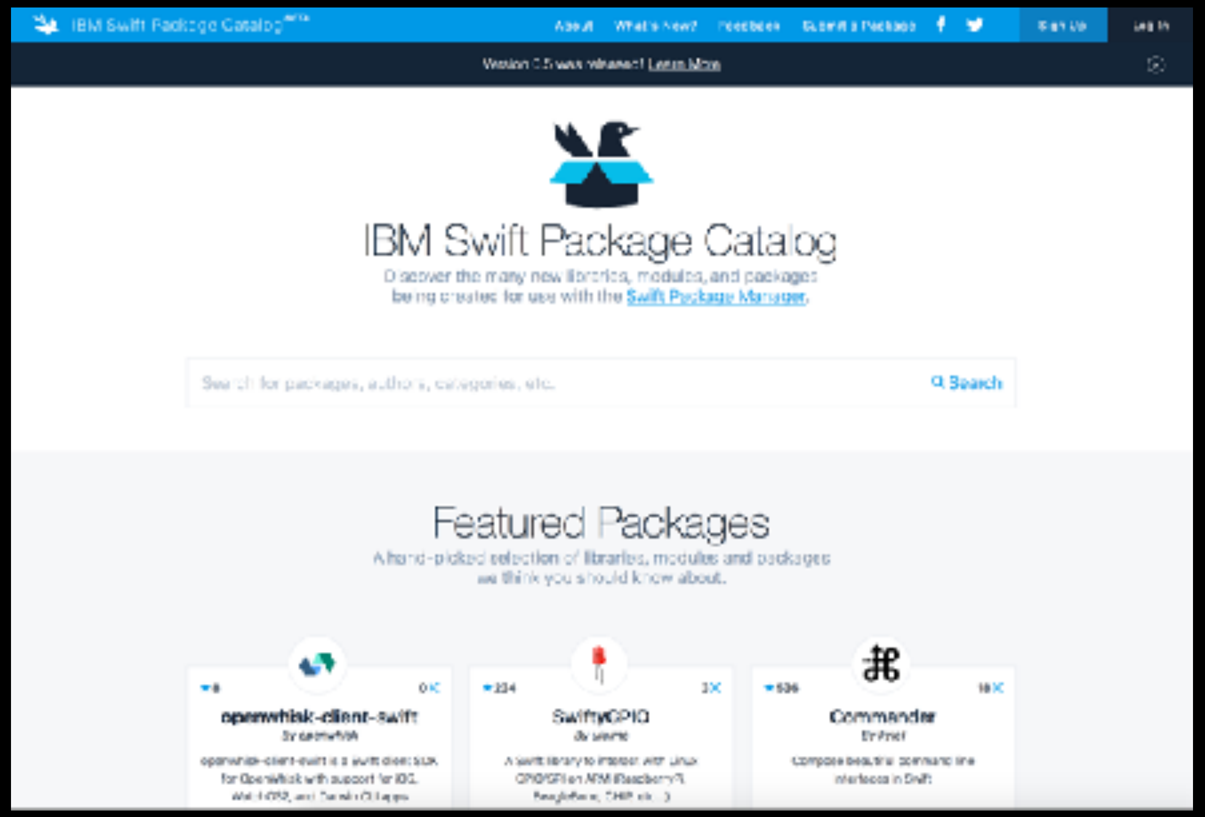


Swift @ IBM

Swift @ **IBM**

https://developer.ibm.com/swift/

Discover

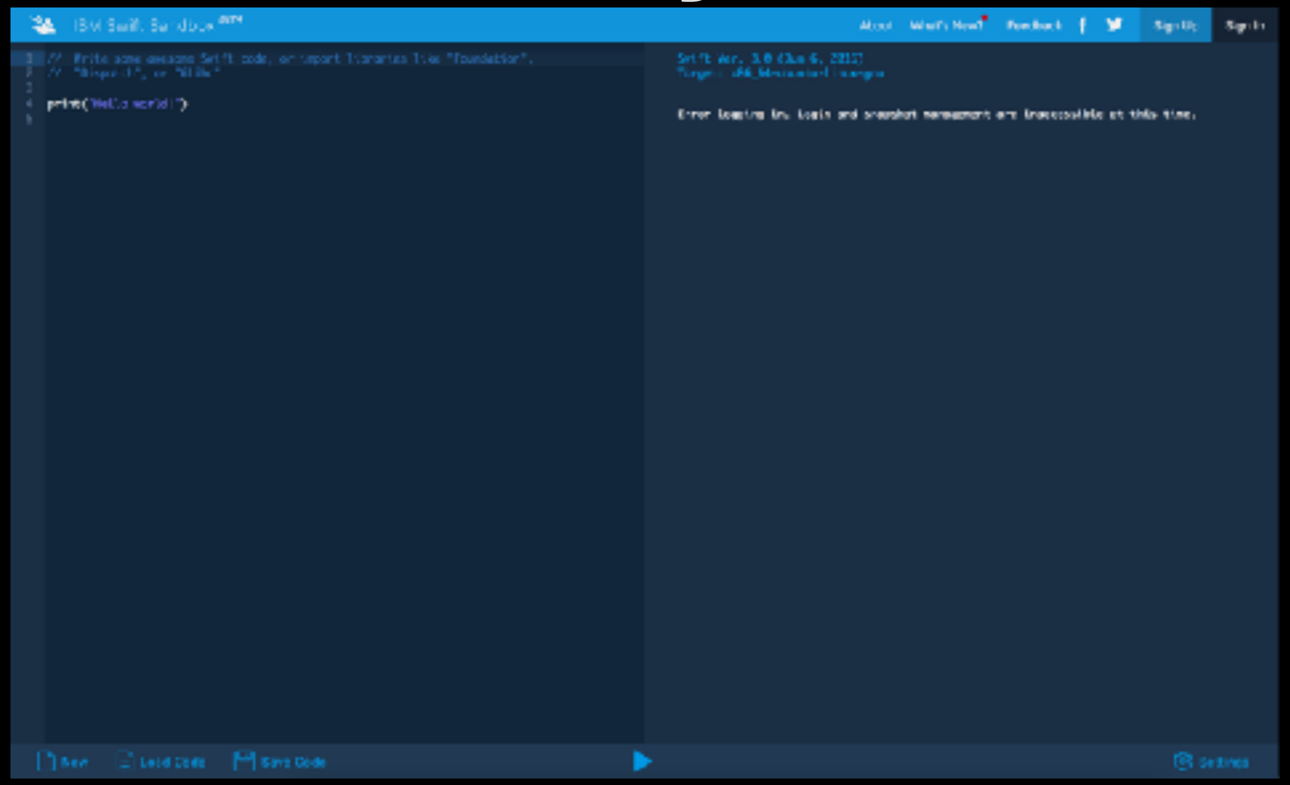

IBM Swift Package Catalog
Discover the many new libraries, modules, and packages being created for use with the Swift Package Manager.

Featured Packages
A hand-picked selection of libraries, modules and packages we think you should know about.

openwhisk-client-swift
SwiftyGPIO
Commander

Package Catalog

Try

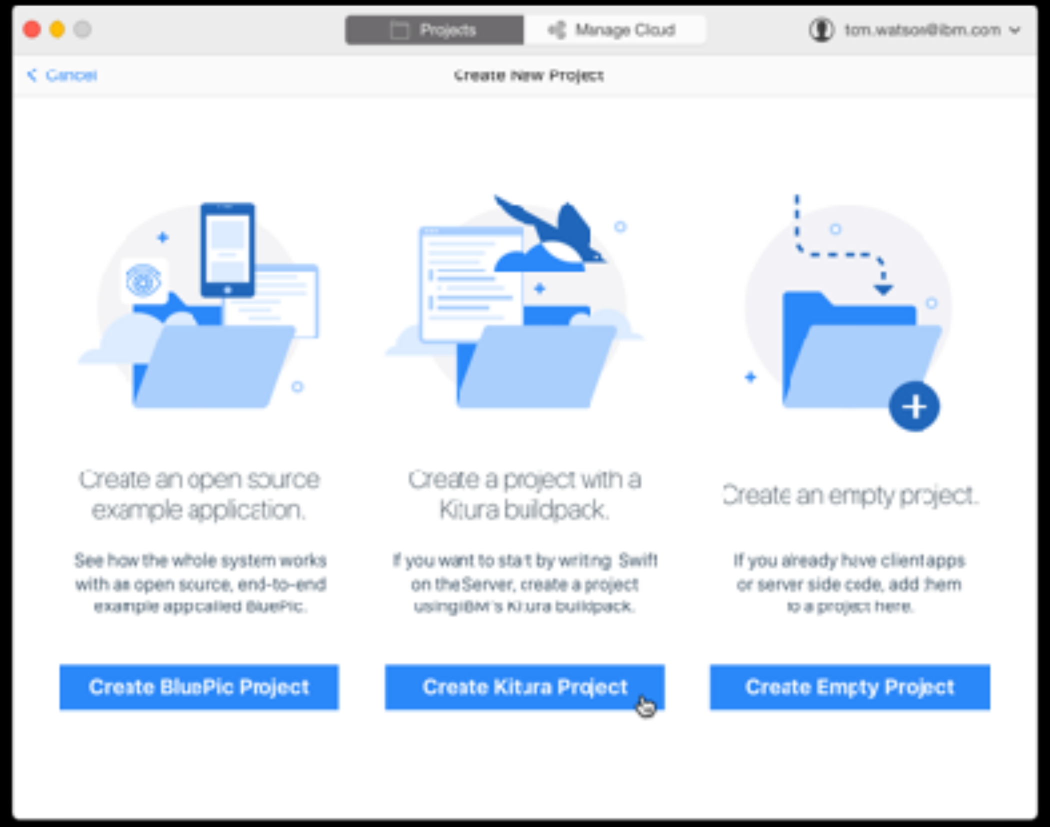


Swift Sandbox

Build

Kitura + Packages

Deploy


Create New Project

Create an open source example application.
See how the whole system works with an open source, end-to-end example app called BluePic.
Create BluePic Project

Create a project with a Kitura buildpack.
If you want to start by writing Swift on the Server, create a project using IBM's Kitura buildpack.
Create Kitura Project

Create an empty project.
If you already have client apps or server side code, add them to a project here.
Create Empty Project

IBM Cloud Tools

Thank you!